(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 August 2010 (05.08.2010)

PCT

(10) International Publication Number
WO 2010/086342 A1

(51) International Patent Classification:
G10L 19/00 (2006.01)

(21) International Application Number:
PCT/EP2010/050954

(22) International Filing Date:
27 January 2010 (27.01.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/147,885 28 January 2009 (28.01.2009) US

(71) Applicant *(for all designated States except US)*: **FRAUNHOFER-GESELLSCHAFT ZUR FÖRDERUNG DER ANGEWANDTEN FORSCHUNG E.V.** [DE/DE]; Hansastraße 27c, 80686 München (DE).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **LECOMTE, Jeremie** [FR/DE]; Sulzbacher Strasse 39, 90489 Nürnberg (DE). **NAGEL, Frederik** [DE/DE]; Wilhelmshavener Str. 72, 90425 Nürnberg (DE). **ROBILLARD, Julien** [FR/DE]; Innerer Kleinreuther Weg 25A, 90408 Nürnberg (DE). **GEIGER, Ralf** [DE/DE]; Maxtorgraben 29, 90409 Nürnberg (DE). **BORSUM, Arne** [DE/DE]; Taeblingstrasse 22, 91058 Erlangen (DE). **FUCHS, Guillaume** [FR/DE]; Fürther Strasse 17, 91058 Erlangen (DE). **MULTRUS, Markus** [DE/DE]; Etzlaubweg 7, 90469 Nürnberg (DE).

(74) Agents: **BURGER, Markus** et al.; Postfach 246, 82043 Pullach (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: AUDIO ENCODER, AUDIO DECODER, METHOD FOR ENCODING AN INPUT AUDIO INFORMATION, METHOD FOR DECODING AN INPUT AUDIO INFORMATION AND COMPUTER PROGRAM USING IMPROVED CODING TABLES

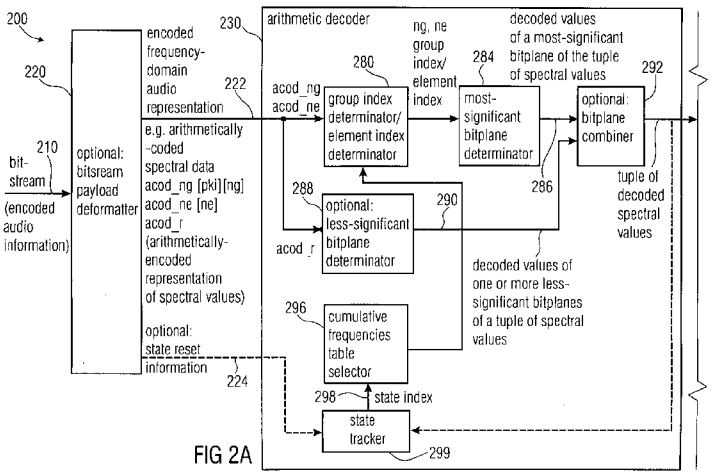


FIG 2A

(57) Abstract: An audio decoder comprises an arithmetic decoder for providing a plurality of decoded spectral values on the basis of an arithmetically-encoded representation of the spectral values and a frequency-domain to time-domain converter for providing a time-domain audio representation using the decoded spectral values, in order to obtain a decoded audio information. The arithmetic decoder is configured to derive a group index from a variable-length-codeword representing the group index in dependence on a state index. The arithmetic decoder is configured to derive the values of a most-significant bit-plane of a tuple of spectral values using the group index and an element index, and to provide a tuple of decoded spectral values using the values of the most-significant bit-plane of the tuple of spectral values. The arithmetic decoder is configured to select a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on the state index, and to apply the selected cumulative-frequencies-table to derive the group index from the variable-length codeword representing the group index.

WO 2010/086342 A1

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

Audio Encoder, Audio Decoder, Method for Encoding an Input Audio Information, Method for Decoding an Input Audio Information and Computer Program using Improved Coding Tables

Background of the Invention

Embodiments according to the invention are related to an audio decoder for providing a decoded audio information on the basis of an encoded audio information, an audio encoder for providing an encoded audio information on the basis of an input audio information, a method for providing a decoded audio information on the basis of an encoded audio information, a method for providing an encoded audio information on the basis of an input audio information and a computer program.

Embodiments according to the invention are related to the concept of using updated arithmetic coder tables in an audio encoder, like, for example, a so-called unified-speech-and-audio-coder (USAC).

In the following, the background of the invention will be briefly explained in order to facilitate the understanding of the invention and the advantages thereof. During the past decade, big efforts have been put on creating the possibility to digitally store and distribute audio contents with good bitrate efficiency. One important achievement on this way is the definition of the International Standard ISO/IEC 14496-3. Part 3 of this Standard is related to an encoding and decoding of audio contents, and subpart 4 of part 3 is related to general audio coding. ISO/IEC 14496 part 3, subpart 4 defines a concept for encoding and decoding of general audio content. In addition, further improvements have been proposed in order to improve the quality and/or to reduce the required bit rate.

According to the concept described in said Standard, a time-domain audio signal is converted into a time-frequency representation. The transform from the time-domain to the time-frequency-domain is typically performed using transform blocks, which are also designated as "frames" of time-domain samples. It has been found that it is advantageous to use overlapping frames, which are shifted, for example, by half a frame, because the overlap allows to efficiently avoid (or at least reduce) artifacts. In addition, it has been found that a windowing should be performed in order to avoid the artifacts originating from this processing of temporally limited frames.

By transforming a windowed portion of the input audio signal from the time-domain to the time-frequency domain, an energy compaction is obtained in many cases, such that some of the spectral values comprise a significantly larger magnitude than a plurality of other spectral values. Accordingly, there are, in many cases, a comparatively small number of spectral values having a magnitude, which is significantly above an average magnitude of the spectral values. A typical example of a time-domain to time-frequency domain transform resulting in an energy compaction is the so-called modified-discrete-cosine-transform (MDCT).

The spectral values are often scaled and quantized in accordance with a psychoacoustic model, such that quantization errors are comparatively smaller for psychoacoustically more important spectral values, and are comparatively larger for psychoacoustically less-important spectral values. The scaled and quantized spectral values are encoded in order to provide a bitrate-efficient representation thereof.

For example, the usage of a so-called Huffman coding of quantized spectral coefficients is described in the International Standard ISO/IEC 14496-3:2005(E), part 3, subpart 4.

However, it has been found that the quality of the coding of the spectral values has a significant impact on the required bitrate. Also, it has been found that the complexity of an audio decoder, which is often implemented in a portable consumer device, and which should therefore be cheap and of low power consumption, is dependent on the coding used for encoding the spectral values.

In view of this situation, there is a need for a concept for an encoding and decoding of an audio content, which provides for an improved trade off between bitrate-efficiency and resource efficiency.

Summary of the Invention

An embodiment according to the invention creates an audio decoder for providing a decoded audio information on the basis of an encoded audio information. The audio decoder comprises an arithmetic decoder for providing a plurality of decoded spectral values on the basis of an arithmetically-encoded representation of the spectral values. The audio decoder also comprises a frequency-domain to time-domain converter for providing a time-domain audio representation using the decoded spectral values, in order to obtain the decoded audio information. The arithmetic decoder is configured to derive, in dependence on a state-index describing a state of the arithmetic decoder, a group-index

from a variable-length codeword representing the group-index. The arithmetic decoder is configured to derive values of a most-significant bit-plane of a tuple of spectral values using the group-index and an element-index, the element index describing (or designating, or selecting) an element within a group selected by the group-index. The arithmetic decoder is configured to provide a tuple of decoded spectral values using the values of the most-significant bit-plane of the tuple of spectral values. The arithmetic decoder is configured to select a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on the state-index describing the state of the arithmetic decoder, and to apply the selected cumulative frequencies-table to derive the group-index from the variable-length-codeword representing the group index.

This embodiment according to the invention is based on the finding that the usage of a set of 32 cumulative-frequencies-tables provides for an optimal trade off between an achievable bitrate and a complexity of an audio encoder or audio decoder. In particular, it has been found that 32 different cumulative-frequencies-tables are appropriate (in that they result in a reasonably low bitrate) for any relevant time-frequency-domain representation of an audio content. It has been found that a set of 32 cumulative-frequencies-tables is optimal, because the usage of a smaller number of cumulative-frequencies-tables would result in a significantly increased bitrate, and because the usage of a larger number of cumulative-frequencies-tables would bring along only insignificant improvements of the bitrate but a remarkable increase of the memory consumption both at the encoder side and at the decoder side.

To summarize the above, an intensive research has brought forward that a number of 32 probability models, which are selected in dependence on a state-index and which are represented by 32 cumulative-frequencies-tables, provides for an optimal trade off between bitrate-efficiency and implementation effort of the arithmetic coding.

In a preferred embodiment, the arithmetic decoder is configured to derive a 7-bit hash-table index from the state-index and to obtain a hash-table entry value from a hash-table, which hash-table comprises a mapping of 128 hash-table index values onto corresponding hash-table entry values. In this case, the arithmetic decoder is configured to decide whether the hash-table entry value (i.e. the content of the hash-table at a memory position designated by the hash-table index value) is an escape value, a valid cumulative-frequencies-table-identifier value associated to a state-index value (e.g. a state-index value on the basis of which the hash-table index value has been derived), or an invalid cumulative-frequencies-table identifier-value (e.g. a cumulative-frequencies-table identifier value which does not

fit the state-index value on the basis of which the hash-table index value has been derived). The arithmetic decoder is configured to scan through the entries of the hash-table until an escape value or a valid cumulative-frequencies-table identifier-value is found. The arithmetic decoder is configured to provide a cumulative-frequencies-index-value in dependence on an identification of an interval of values in which the state-index value is contained, if the obtained hash-table entry value is the escape value, and to derive the cumulative-frequencies-index-value from the obtained hash-table entry value, if the hash-table entry value is a valid cumulative-frequencies-table identifier-value associated to the state-index value.

This embodiment of the invention is based on the finding that there are only a small number of "significant" states of the audio decoder, for which it is important to use a specific probability model (associated to a small number of only about 1 to 10 states), represented by a specific cumulative-frequencies-table (associated to a small number of only about 1 to 10 states). It has also been found that for the vast majority of states of the audio decoder, it is preferred to map the states onto a cumulative-frequencies-index-value on the basis of a determination of an interval of values, in which the state-index is contained, thereby mapping a whole interval of state-index values (typically a range of more than 1000 different state-index values) onto the same cumulative-frequencies-index-value.

As a consequence of the finding that a bitrate-efficient encoding can be obtained even if the number of "significant" states, to which specific probability distributions are associated, is very small, it has been possible to reduce the size of the hash-table to only 128 entries (while hash tables more than 100 times larger were used previously). Accordingly, the encoder-sided and decoder-sided implementation of the hash-table requires a very small amount of resources. This helps to make audio decoders cheap and also to keep the energy consumption of such audio decoders reasonably small, which in turn improves the possibilities to implement cheap and mobile consumer devices.

In a preferred embodiment, the hash-table is configured to map 67 values of the 7-bit hash-table-index onto valid cumulative-frequencies-table identifier-values and to map 61 values of the 7-bit hash-table-index onto escape values. Accordingly, a comparatively small number of only 67 "significant" states are present. In addition, any "insignificant" states (the number of which insignificant states is significantly larger than the number of significant states, for example by a factor of at least 10, but preferably by a factor of even more than 1000), which differ from the "significant" states, are mapped onto escape

values, thereby making it easy to distinguish between a "significant" state and an "insignificant" state. Accordingly, the difference between significant and insignificant states can be determined fast and with low memory consumption.

In a preferred embodiment, the arithmetic decoder is configured to map 67 different values of the state index onto 67 different cumulative-frequencies-table identifier-values, such that 26 different cumulative-frequencies index values are associated to 67 different significant states described by the state-index value.

It has been found that even for the comparatively low number of 67 different significant states, it is sufficient to have only 26 different cumulative-frequencies distributions associated therewith. Again, the choice of only 26 different cumulative-frequencies-tables associated to the 67 significant states has shown to bring along an optimum trade-off between bitrate requirements and encoding/decoding complexity requirements.

In a preferred embodiment, the arithmetic decoder is configured to map the different non-significant states onto nine different cumulative-frequencies index values, such that a total of nine different cumulative-frequencies-tables are available for use in connection with the non-significant states, for which a range-type mapping is performed in order to derive the cumulative-frequencies index value. It has been found that a very small number of preferably nine different cumulative-frequencies-tables is sufficient in order to obtain a bitrate-efficient arithmetic encoding for most of the states.

It has also been found that there is a strong difference between a comparatively small number of significant states and a large number of non-significant states. The preferably 67 significant states, to which cumulative-frequencies-tables are associated by valid cumulative-frequencies-table identifier values of the hash table, are associated with particularly characteristic patterns of the spectral values like, for example, trajectories in the time-frequency representation having a particular width and direction. In contrast, all the other states, which are considered as non-significant states and to which a cumulative-frequencies-table is associated using a range-based algorithm, merely represent less characteristic spectral value distributions.

Notably, it has also been found that some of the cumulative-frequencies-tables, which are associated to the non-significant states, are also well-suited for being used in connection with some of the significant states. Accordingly, it has been found that it is advantageous to use three cumulative-frequencies-tables (e.g. cumulative-frequencies-tables having

associated therewith indices 05, 26 and 30) which are applied both for one or more significant states and also for one or more non-significant states. For example, it has been found to be particularly advantageous, in terms of a trade-off between bitrate-efficiency and computational complexity, to have 23 different cumulative-frequencies-tables which are associated to significant states only using valid cumulative-frequencies-table identifier values of the hash table, to have six cumulative-frequencies-tables which are associated to non-significant states only using an escape-mechanism, which escape mechanism is based on an escape value stored in the hash table, and a range-based mapping. Also, it has been found to be advantageous to have three cumulative-frequencies-tables, which are associated both to one or more of the significant states and to one or more of the non-significant states.

Another embodiment according to the invention creates an audio encoder for providing an encoded audio information on the basis of an input audio information. The audio encoder comprises a time-domain to frequency-domain converter for providing a frequency-domain audio representation on the basis of a time-domain representation of the input audio information, such that the frequency-domain audio representation comprises a set of spectral values. The audio encoder also comprises an arithmetic encoder configured to encode a tuple of adjacent spectral values, or a preprocessed version thereof, using a variable-length codeword. The arithmetic encoder is configured to map values of a most-significant bit plane of a tuple of spectral values onto a group index and an element index, the element index describing an element within a group selected by the group index. The arithmetic encoder is further configured to select a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on a state index describing a state of the arithmetic encoder and to arithmetically encode the group index using a selected cumulative-frequencies-table, in order to obtain an arithmetically encoded variable-length codeword.

The audio encoder according to this embodiment is based on the same idea as the audio decoder discussed above. In particular, the audio encoder is based on the finding that the number of 32 cumulative-frequencies-tables brings along an optimized trade-off between bitrate-efficiency and encoding/decoding complexity.

Another embodiment according to the invention creates a method for providing a decoded audio representation on the basis of an encoded audio representation.

Another embodiment according to the invention creates a method for providing an encoded audio representation on the basis of an input audio representation.

Another embodiment according to the invention creates a computer program for performing the inventive methods.

Brief Description of the Figures

Embodiments according to the present invention will subsequently be described taking reference to the enclosed figures, in which:

Fig. 1 shows a block schematic diagram of an audio encoder, according to an embodiment of the invention;

Fig. 2 shows a block schematic diagram of an audio decoder, according to an embodiment of the invention;

Fig. 3 shows a pseudo-program-code representation of an algorithm "tuples_decode()" for decoding a tuple of spectral values;

Fig. 4 shows a schematic representation of a context for a state calculation;

Fig. 5a shows a pseudo-program-code representation of an algorithm "arith_reset_context ()" for resetting a context;

Fig. 5b shows a pseudo-program-code representation of an algorithm "arith_map_context ()" for mapping a context;

Fig. 5c shows a pseudo-program-code representation of an algorithm "arith_get_context ()" for obtaining a context state value;

Fig. 5d shows a pseudo-program-code representation of an algorithm "arith_get_pk(s)" for deriving a cumulative-frequencies-table index value pki from a state variable;

Fig. 5e shows a pseudo-program-code representation of an algorithm "arith_decode ()" for arithmetically decoding a symbol from a variable-length codeword;

Fig. 5f shows a pseudo-program-code representation of an algorithm for deriving an element number value mm and a group offset value og from a group index nq;

Fig. 5g shows a pseudo-program-code representation of an algorithm for obtaining spectral values a, b, c, d of a most-significant bit-plane of a tuple of spectral values on the basis of the group offset value og and an element index value ne;

Fig. 5h shows a pseudo-program-code representation of an algorithm for combining a tuple a, b, c, d of spectral values with values of a less-significant bit-plane, in order to obtain an updated version of the tuple a, b, c, d of spectral values;

Fig. 5i shows a pseudo-program-code representation of an algorithm "arith_update_context ()" for updating the context;

Fig. 5j shows a legend of definitions and variables;

Fig. 6a shows as syntax representation of a unified-speech-and-audio-coding (USAC) raw data block;

Fig. 6b shows a syntax representation of a single channel element;

Fig. 6c shows syntax representation of a channel pair element;

Fig. 6d shows a syntax representation of an "ics" control information;

Fig. 6e shows a syntax representation of a frequency-domain channel stream;

Fig. 6f shows a syntax representation of arithmetically-coded spectral data;

Fig. 6g shows a syntax representation for decoding a set of tuples of spectral values;

Fig. 6h shows a legend of data elements and variables;

Fig. 7 shows a table representation of memory requirements of a previously used arithmetic coder;

Fig. 8 shows a table representation of memory requirements of an arithmetic coder according to the present invention;

Fig. 9 shows a block schematic diagram of a set-up for evaluating performance improvements obtained by the arithmetic coder according to the present invention;

Fig. 10 shows a table representation of bitrates required for encoding different pieces of audio information using a previously used arithmetic coder;

Fig. 11 shows a table representation of bitrates required for encoding different pieces of audio information using the inventive concept;

Fig. 12 shows, in the form of a table representation, a comparison between average bitrates produced by a previously used audio coder and an audio coder according to the present invention;

Fig. 13 shows, in the form of a table representation, a comparison of bitrate reductions and bitrate increases obtained using the inventive concept when compared to a previously used concept;

Fig. 14 shows a table representation of entries of a table "arith_cf_ng_hash[]";

Figs. 15(1) to 15(10) show a table representation of entries of a table "arith_cf_ne []";

Figs. 16(1) to 16(32) show a table representation of entries of a table "arith_cf_ng [pki] [545]" for 32 different values 0 to 31 of the index pki;

Figs. 17(1) and 17(2) show a table representation of entries of a table "dgroups []";

Figs. 18(1) to 18(11) show a table representation of entries of a table "dvectors[]" ;

Figs. 19(1) to 19(32) show a table representation of entries of a table "egroups [a][b][c][d]";

Fig. 20 show a flow chart of a method for providing a decoded representation of an audio information; and

Fig. 21 shows a flow chart of a method for providing an encoded representation of an audio information.

Detailed Description of the Embodiments

1. Audio Encoder

In the following, an audio encoder according to an embodiment of the present invention will be described. Fig. 1 shows a block schematic diagram of such an audio encoder 100.

The audio encoder 100 is configured to receive an input audio information 110 and to provide, on the basis thereof, a bitstream 112, which constitutes an encoded audio information. The audio encoder 100 optionally comprises a preprocessor 120, which is configured to receive the input audio information 110 and to provide, on the basis thereof, a preprocessed input audio information 110a. The audio encoder 100 also comprises an energy-compacting time-domain to frequency-domain signal transformer 130, which is also designated as signal converter. The signal converter 130 is configured to receive the input audio information 110, 110a and to provide, on the basis thereof, a frequency-domain audio information 132, which preferably takes the form of a set of spectral values. For example, the signal transformer 130 may be configured to receive a frame of the input audio information 110, 110a (for example, a block of time-domain samples) and to provide a set of spectral values representing the audio content of the respective audio frame. In addition, the signal transformer 130 may be configured to receive a plurality of subsequent, overlapping or non-overlapping, audio frames of the input audio information 110, 110a and to provide, on the basis thereof, a time-frequency-domain audio representation, which comprises a sequence of subsequent sets of spectral values, one set of spectral values associated with each frame.

The energy-compacting time-domain to frequency-domain signal transformer 130 may comprise an energy-compacting filterbank, which provides spectral values associated with different, overlapping or non-overlapping, frequency ranges. For example, the signal transformer 130 may comprise a windowing MDCT transformer 130a, which is configured to window the input audio information 110, 110a (or a frame thereof) using a transform window and to perform a modified-discrete-cosine-transform of the windowed input audio information 110, 110a (or of the windowed frame thereof). Accordingly, the frequency-

domain audio representation 132 may comprise a set of, for example, 1024 spectral values in the form of MDCT coefficients associated with a frame of the input audio information.

The audio encoder 100 may further, optionally, comprise a spectral post-processor 140, which is configured to receive the frequency-domain audio representation 132 and to provide, on the basis thereof, a post-processed frequency-domain audio representation 142. The spectral post-processor 140 may, for example, be configured to perform a temporal noise shaping and/or a long term prediction and/or any other spectral post-processing known in the art. The audio encoder further comprises, optionally, a scaler/quantizer 150, which is configured to receive the frequency-domain audio representation 132 or the post-processed version 142 thereof and to provide a scaled and quantized frequency-domain audio representation 152.

The audio encoder 100 further comprises, optionally, a psycho-acoustic model processor 160, which is configured to receive the input audio information 110 (or the post-processed version 110a thereof) and to provide, on the basis thereof, an optional control information, which may be used for the control of the energy-compacting time-domain to frequency-domain signal transformer 130, for the control of the optional spectral post-processor 140 and/or for the control of the optional scaler/quantizer 150. For example, the psycho-acoustic model processor 160 may be configured to analyze the input audio information, to determine which components of the input audio information 110, 110a are particularly important for the human perception of the audio content and which components of the input audio information 110, 110a are less important for the perception of the audio content. Accordingly, the psycho-acoustic model processor 160 may provide control information, which is used by the audio encoder 100 in order to adjust the scaling of the frequency-domain audio representation 132, 142 by the scaler/quantizer 150 and/or the quantization resolution applied by the scaler/quantizer 150. Consequently, perceptually important scale factor bands (i.e. groups of adjacent spectral values which are particularly important for the human perception of the audio content) are scaled with a large scaling factor and quantized with comparatively high resolution, while perceptually less-important scale factor bands (i.e. groups of adjacent spectral values) are scaled with a comparatively smaller scaling factor and quantized with a comparatively lower quantization resolution. Accordingly, scaled spectral values of perceptually more important frequencies are typically significantly larger than spectral values of perceptually less important frequencies.

The audio encoder also comprises an arithmetic encoder 170, which is configured to receive the scaled and quantized version 152 of the frequency-domain audio representation

132 (or, alternatively, the post-processed version 142 of the frequency-domain audio representation 132, or even the frequency-domain audio representation 132 itself) and to provide arithmetic codeword information 172a, 172b on the basis thereof, such that the arithmetic codeword information represents the frequency-domain audio representation 152.

The audio encoder 100 also comprises a bitstream payload formatter 190, which is configured to receive the arithmetic codeword information 172a, 172b. The bitstream payload formatter 190 is also typically configured to receive additional information, like, for example, scale factor information describing which scale factors have been applied by the scaler/quantizer 150. In addition, the bitstream payload formatter 190 may be configured to receive other control information. The bitstream payload formatter 190 is configured to provide the bitstream 112 on the basis of the received information by assembling the bitstream in accordance with a desired bitstream syntax, which will be discussed below.

In the following, details regarding the arithmetic encoder 170 will be described. The arithmetic encoder 170 is configured to receive a plurality of tuples of, for example, four post-processed and scaled and quantized spectral values of the frequency-domain audio representation 132. The arithmetic encoder comprises a most-significant-bitplane-extractor 174, which is configured to extract a most-significant bit-plane from a tuple of spectral values. It should be noted here that the most-significant bit-plane may comprise one or even more bits (for example, two or three bits), which are the most-significant bits of the spectral values of the tuple of spectral values. Thus, the most-significant bit-plane extractor 174 provides a most-significant bit-plane 176 of a tuple of spectral values (which spectral values are preferably adjacent in frequency). The arithmetic encoder 170 also comprises a group index determinator/element index determinator 178, which is configured to map the most-significant bit-plane 176 onto a group-index value ng and an element-index value ne. This mapping may be performed using a look-up table, for example, the look-up table "egroups" discussed in detail below. The group index determinator/element index determinator 178 may be configured to map some of the combinations of values of the most-significant bit-plane 176 onto a group index ng of a group comprising only one element and may be configured to map other combinations of values of the most-significant bit-plane 176 onto a group index ng of a group comprising a plurality of combinations of values. Accordingly, the group index determinator/element index determinator may be configured to map such combinations of values of the most-significant bit-plane 176 which comprise a comparatively high probability onto groups

comprising only one or only a few elements, and to map combinations of values of the most-significant bit-plane 176 comprising a comparatively lower probability onto groups comprising more elements. Consequently, the element index ne of a combination of values, which is mapped to a group comprising only a single element, may only take a single value and may therefore be neglected. In contrast, the element index ne of a combination of values, which is mapped to a group comprising a plurality of elements, may take a plurality of values. Thus, the group index determinator/element index determinator 178 provides a group index value ng (also designated with 180a) and, if required, the element index value ne (also designated with 180b), wherein the element index value ne may be set to a default value or omitted if the group ng to which the most-significant bit- plane 176 is mapped, comprises a single element only.

The arithmetic encoder 170 also comprises a first codeword determinator 180, which is configured to determine an arithmetic codeword acod_ng [pki][ng] representing the group index ng. In addition, the first codeword determinator 180 may provide an arithmetic codeword acod_ne [ne] representing the element index ne, if the number of elements mm of the group ng is larger than 1. Otherwise, the provision of the arithmetic codeword acod_ne [ne] representing the element index ne may be omitted. Optionally, the codeword determinator 180 may also provide one or more escape codewords (also designated herein with "ARITH_ESCAPE") indicating, for example, how many less-significant bit-planes are available (and, consequently, indicating the numeric weight of the most-significant bit-plane). The first codeword determinator 180 may be configured to provide the codeword associated with a group index ng using a selected cumulated-frequencies-table having (or being referenced by) a cumulative-frequencies-table index pki.

In order to determine as to which cumulative-frequencies-table should be selected, the arithmetic encoder preferably comprises a state tracker 182, which is configured to track the state of the arithmetic encoder, for example, by observing which tuples of spectral values have been encoded previously. The state tracker 182 consequently provides a state information 184, for example a state value designated with "s" or "t". The arithmetic encoder 170 also comprises a cumulative-frequencies-table selector 186, which is configured to receive the state information 184 and to provide an information 188 describing the selected cumulative-frequencies-table to the codeword determinator 180. For example, the cumulative-frequencies-table selector 186 may provide a cumulative-frequencies-table index pki describing which cumulative-frequencies-table, out of a set of 32 cumulative frequencies tables, is selected for usage by the codeword determinator. Alternatively, the cumulative-frequencies-table selector 186 may provide the entire selected cumulative-frequencies-table to the codeword determinator. Thus, the codeword

determinator 180 may use the selected cumulative-frequencies-table for the provision of the codeword acod_ng[pki][ng] of the group index ng, such that the actual codeword acod_ng[pki][ng] encoding the group index ng is dependent on the value of ng and the cumulative-frequencies-table index pki, and consequently on the current state information 184. In contrast, the first codeword determinator 180 may use a default (state-independent) cumulative-frequencies-table for the provision of the codeword acod_ne [ne], which may, however, be dependent on the number of elements within the selected group ng. Further details regarding the coding process and the obtained codeword format will be described below.

The arithmetic encoder 170 further comprises a less-significant bit-plane extractor 189a, which is configured to extract one or more less-significant bit-planes from the scaled and quantized frequency-domain audio representation 152, if one or more of the values of a tuple of spectral values to be encoded exceed the range of values encodeable using the most-significant bit-plane only. The less-significant bit-planes may comprise one or more bits per spectral values, as desired. Accordingly, the less-significant bit-plane extractor 189a provides a less-significant bit-plane information 189b. The arithmetic encoder 170 also comprises a second codeword determinator 189c, which is configured to receive the less-significant bit-plane information 189d and to provide, on the basis thereof, 0, 1 or more codewords "acod_r" representing the content of 0, 1 or more less-significant bit-planes. The second codeword determinator 189c may be configured to apply an arithmetic encoding algorithm or any other encoding algorithm in order to derive the less-significant bit-plane codewords "acod_r" from the less-significant bit-plane information 189b.

It should be noted here that the number of less-significant bit-planes may vary in dependence on the value of the scaled and quantized spectral values 152, such that there may be no less-significant bit-plane at all, if the scaled and quantized spectral values of the current tuple are comparatively small, such that there may be one less-significant bit-plane if the scaled and quantized spectral values of the current tuple are of a medium range and such that there may be more than one less-significant bit-plane if the scaled and quantized spectral values take a comparatively large value.

To summarize the above, the arithmetic encoder 170 is configured to encode a tuple of scaled and quantized spectral values, which is described by the information 152, using a hierarchical encoding process. The most-significant bit-plane (comprising, for example, one, two or three bits per spectral value) is encoded to obtain an arithmetic codeword "acod_ng [pki][ng]" of a group index ng and, in some cases, a codeword "acod_ne [ne]" of an element index ne. One or more less-significant bit-planes (each of the less-significant

bit-planes comprising, for example, one, two or three bits) are encoded to obtain one or more codewords "acod_r". When encoding the most-significant bit-plane, the combination of values of the most-significant bit-plane is mapped to a group ng of the plurality of groups wherein some of the groups comprise one element only and wherein others of the groups comprise a plurality of elements each. Accordingly, the probability of the different combinations of values is considered. Subsequently, the group index ng and the element index ne (if required) are encoded, wherein 32 different cumulative-frequencies-tables are available for the encoding of the group index ng in dependence on a state of the arithmetic encoder 170, i.e. in dependence on previously-encoded tuples of spectral values. Accordingly, codewords "acod_ng [pki][ng]" and "acod_ne [ne]" are obtained, wherein the latter codeword is only included in the bitstream 112 if the group index ng designates a group comprising more than one element. In addition, one or more codewords "acod_r" are provided and included into the bitstream if one or more less-significant bit-planes are present.

Reset description

The audio encoder 100 may optionally be configured to decide whether an improvement in bitrate can be obtained by resetting the context, for example by setting the state index to a default value. Accordingly, the audio encoder 100 may be configured to provide a reset information (e.g. named "arith_reset_flag") indication whether the context for the arithmetic encoding is reset, and also indicating whether the context for the arithmetic decoding in a corresponding decoder should be reset.

Details regarding the bitstream format and the applied cumulative-frequency tables will be discussed below.

2. Audio Decoder

In the following, an audio decoder according to an embodiment of the invention will be described. Fig. 2 shows a block schematic diagram of such an audio decoder 200.

The audio decoder 200 is configured to receive a bitstream 210, which represents an encoded audio information and which may be identical to the bitstream 112 provided by the audio encoder 100. The audio decoder 200 provides a decoded audio information 212 on the basis of the bitstream 210.

The audio decoder 200 comprises an optional bitstream payload deformatter 220, which is configured to receive the bitstream 210 and to extract from the bitstream 210 an encoded frequency-domain audio representation 222. For example, the bitstream payload deformatter 220 may be configured to extract from the bitstream 210 arithmetically-coded spectral data like, for example, an arithmetic codeword "acod_ng [pki][ng]" representing a group index ng, an arithmetic codeword "acod_ne [ne]" representing an element index ne and a codeword "acod_r" representing a content of a less-significant bit-plane of the frequency-domain audio representation. Thus, the encoded frequency-domain audio representation 222 constitutes (or comprises) an arithmetically-encoded representation of spectral values. The bitstream payload deformatter 220 is further configured to extract from the bitstream additional control information, which is not shown in Fig. 2. In addition, the bitstream payload deformatter is optionally configured to extract from the bitstream 210 a state reset information 224, which is also designated as arithmetic reset flag or "arith_reset_flag".

The audio decoder 200 comprises an arithmetic decoder 230, which is also designated as "spectral noiseless decoder". The arithmetic decoder 230 is configured to receive the encoded frequency-domain audio representation 220 and, optionally, the state reset information 224. The arithmetic decoder 230 is also configured to provide a decoded frequency-domain audio representation 232, which may comprise a decoded representation of spectral values. For example, the decoded frequency-domain audio representation 232 may comprise a decoded representation of tuples of spectral values, which are described by the encoded frequency-domain audio representation 220.

The audio decoder 200 also comprises an optional inverse quantizer/rescaler 240, which is configured to receive the decoded frequency-domain audio representation 232 and to provide, on the basis thereof, an inversely-quantized and rescaled frequency-domain audio representation 242.

The audio decoder 200 further comprises an optional spectral pre-processor 250, which is configured to receive the inversely-quantized and rescaled frequency-domain audio representation 242 and to provide, on the basis thereof, a pre-processed version 252 of the inversely-quantized and rescaled frequency-domain audio representation 242. The audio decoder 200 also comprises a frequency-domain to time-domain signal transformer 260, which is also designated as a "signal converter". The signal transformer 260 is configured to receive the pre-processed version 252 of the inversely-quantized and rescaled frequency-domain audio representation 242 (or, alternatively, the inversely-quantized and

rescaled frequency-domain audio representation 242 or the decoded frequency-domain audio representation 232) and to provide, on the basis thereof, a time-domain representation 262 of the audio information. The frequency-domain to time-domain signal transformer 260 may, for example, comprise a transformer for performing an inverse-modified-discrete-cosine transform (IMDCT) and an appropriate windowing (as well as other auxiliary functionalities, like, for example, an overlap-and-add).

The audio decoder 200 may further comprise an optional time-domain post-processor 270, which is configured to receive the time-domain representation 262 of the audio information and to obtain the decoded audio information 212 using a time-domain post-processing. However, if the post-processing is omitted, the time-domain representation 262 may be identical to the decoded audio information 212.

It should be noted here that the inverse quantizer/rescaler 240, the spectral pre-processor 250, the frequency-domain to time-domain signal transformer 260 and the time-domain post-processor 270 may be controlled in dependence on control information, which is extracted from the bitstream 210 by the bitstream payload deformatter 220.

To summarize the overall functionality of the audio decoder 200, a decoded frequency-domain audio representation 232, for example, a set of spectral values associated with an audio frame of the encoded audio information, may be obtained on the basis of the encoded frequency-domain representation 222 using the arithmetic decoder 230. Subsequently, the set of, for example, 1024 spectral values, which may be MDCT coefficients, are inversely quantized, rescaled and pre-processed. Accordingly, an inversely-quantized, rescaled and spectrally pre-processed set of spectral values (for example, 1024 MDCT coefficients) is obtained. Afterwards, a time-domain representation of an audio frame is derived from the inversely-quantized, rescaled and spectrally pre-processed set of frequency-domain values (e.g. MDCT coefficients). Accordingly, a time-domain representation of an audio frame is obtained. The time-domain representation of a given audio frame may be combined with time-domain representations of previous and/or subsequent audio frames. For example, an overlap-and-add between time-domain representations of subsequent audio frames may be performed in order to smoothen the transitions between the time-domain representations of the adjacent audio frames and in order to obtain an aliasing cancellation. For details regarding the reconstruction of the decoded audio information 212 on the basis of the decoded time-frequency domain audio representation 232, reference is made, for example, to the International Standard ISO/IEC 14496-3, part 3, sub-part 4 where a detailed discussion is given.

In the following, some details regarding the arithmetic decoder 230 will be described. The arithmetic decoder 230 comprises a group index determinator/element index determinator 280, which is configured to receive the arithmetic codeword acod_ng [pki][ng] describing the group index ng and to also receive the codeword acod_ne [ne] of the element index ne if the codeword acod_ne [ne] is available. The group index determinator 280 is configured to provide a decoded group index value ng and to also provide a decoded element index value ne if the group described by the group index value ng comprises more than one element. However, the group index determinator/element index determinator 280 may be configured to provide the default element index value ne, for example, of one if the group described by the group index value ng comprises one element only. The group index determinator/element index determinator 280 may be configured to use a cumulative-frequencies table out of a set comprising a plurality of 32 cumulative frequencies tables for deriving the group index value ng from the arithmetic codeword "acod_ng [pki][ng]". The arithmetic decoder 280 further comprises a most-significant bit-plane determinator 284, which is configured to derive values 286 of a most-significant bit-plane of a 2-bit tuple (or 3-bit tuple) of spectral values on the basis of a group index value ng and an element index value ne. The arithmetic decoder 230 further comprises a less-significant bit-plane determinator 288, which is configured to receive one or more codewords "acod_r" representing one or more less-significant bit-planes of a tuple of spectral values. Accordingly, the less-significant bit-plane determinator 288 is configured to provide decoded values 290 of one or more less-significant bit-planes. The audio decoder 200 also comprises a bit-plane combiner 292, which is configured to receive the decoded values 286 of the most-significant bit-plane of the tuple of spectral values and the decoded values 290 of one or more less-significant bit planes of the tuple of spectral values if such less-significant bit-planes are available for the current tuple of spectral values. Accordingly, the bit-plane combiner 292 provides a tuple of encoded spectral values, which is part of the decoded frequency-domain audio representation 232. Naturally, the arithmetic decoder 230 is typically configured to provide a plurality of tuples of decoded spectral values in order to obtain a full set of decoded spectral values associated with a current frame of the audio content.

The arithmetic decoder 230 further comprises a cumulative-frequencies-table selector 296, which is configured to select one of the 32 cumulative-frequencies tables in dependence on a state index 298 describing a state of the arithmetic decoder. The arithmetic decoder 230 further comprises a state tracker 299, which is configured to track a state of the arithmetic decoder in dependence on the tuples of previously-decoded spectral values. The state information may optionally be reset to a default state information in response to the state reset information 224. Accordingly, the cumulative-frequencies-table selector 296 is

configured to provide an index (e.g. pki) of a selected cumulative-frequencies-table, or a selected cumulative-frequencies-table itself, to the group index determinator/element index determinator 280 for application in the decoding of the group index ng in dependence on the group index codeword "acod_ng".

To summarize the functionality of the audio decoder 200, the audio decoder 200 is configured to receive a bitrate-efficiently-encoded frequency-domain audio representation 222 and to obtain a decoded frequency-domain audio representation on the basis thereof. In the arithmetic decoder 230, which is used for obtaining the decoded frequency-domain audio representation 232 on the basis of the encoded frequency-domain audio representation 222, a probability of different combinations of values of the most-significant bit-plane is exploited by using an arithmetic decoder 280, which is configured to apply a cumulative-frequencies-table. In addition, statistic dependencies between different tuples of spectral values are exploited by selecting different cumulative-frequencies-tables out of a set comprising 32 different cumulative-frequencies-tables in dependence on a state index 298, which is obtained by observing the previously-computed tuples of decoded spectral values.

3. Overview over the Tool of Spectral Noiseless Coding

In the following, details regarding the encoding and decoding algorithm, which is performed, for example, by the arithmetic encoder 170 and the arithmetic decoder 230 will be explained.

Focus is put on the description of the decoding algorithm. It should be noted, however, that a corresponding encoding algorithm can be performed in accordance with the teachings of the decoding algorithm, wherein mappings are inversed.

It should be noted that the decoding, which will be discussed in the following, is used in order to allow for a so-called "spectral noiseless coding" of typically post-processed, scaled and quantized spectral values. The spectral noiseless coding is used in an audio encoding/decoding concept to further reduce the redundancy of the quantized spectrum, which is obtained, for example, by an energy-compacting time-domain to a frequency-domain transformer.

The spectral noiseless coding scheme, which is used in embodiments of the invention, is based on an arithmetic coding in conjunction with a dynamically-adapted context. In the

preferred embodiment and in the following, the spectral values are processed by the spectral noiseless coding by tuples combining 4 successive spectral values in frequency and called then 4-tuples. The noiseless coding is fed by (original or encoded representations of) quantized spectral values and uses context-dependent cumulative-frequencies-tables derived, for example, from four previously-decoded neighboring 4-tuples. Here, the neighborhood in both time and frequency is taken into account as illustrated in Fig. 4. The cumulative-frequencies-tables (which will be explained below) are then used by the arithmetic coder to generate a variable-length binary code and by the arithmetic decoder to derive decoded values from a variable-length binary code.

For example, the arithmetic coder 170 produces a binary code for a given set of symbols in dependence on the respective probabilities. The binary code is generated by mapping a probability interval, where the set of symbol lies, to a codeword.

4. Decoding Process

4.1 Decoding Process Overview

In the following, an overview of the process of decoding a tuple of spectral values will be given taking reference to Fig. 3, which shows a pseudo-program code representation of the process of decoding a plurality of tuples of spectral values.

The process of decoding a plurality of tuples of spectral values comprises an initialization 310 of a context. The initialization 310 of the context selectively comprises a reset of the context using the function "arith_reset_context ()" or a derivation of the current context from a previous context using the function "arith_map_context (lg/4)". Both the reset of the context and the derivation of the current context from a previous context will be discussed below.

The decoding of a plurality of tuples of spectral values also comprises an iteration of a tuple decoding 312 and a context update 314, which context update is performed by a function "arith_update_context (a,b,c,d,i,lg/4)", which is described below. The tuple decoding 312 and the context update 314 are repeated lg/4 times, wherein lg/4 indicates the number of tuples of spectral values to be decoded. The tuple decoding 312 comprises a context-value calculation 312a, a group index decoding 312b, an element index decoding 312c, a most-significant bit-plane determination 312d and a less-significant bit-plane addition 312e.

The state value computation 312a comprises the computation of a first state value s using the function "arith_get_context (i)", which function returns the first state value s. The state value computation 312a also comprises a computation of a level value lev, which is obtained by shifting the first state value s to the right by 24-bits. The state value computation 312a also comprises a computation of a second state value t according to the formula shown in Fig. 3.

The group index decoding 312b comprises an iterative execution of a decoding algorithm 312ba, wherein a variable j is initialized to 0 before a first execution of the algorithm 312ba.

The algorithm 312ba comprises a computation of a state index pki (which also serves as a cumulative-frequencies-table index) in dependence on the second state value t using a function "arith_get_pk()", which is discussed above. The algorithm 312ba also comprises the selection of a cumulative-frequencies-table in dependence on the state index pki, wherein a variable "cum_freq" may be set to a starting address of one out of 32 cumulative-frequencies-tables in dependence on the state index pki. Also, a variable "cfl" may be initialized to a length of the selected cumulative-frequencies-table, which is equal to the number of symbols in the alphabet, i.e. the number of different value which cane be decoded. The lengths of all the cumulative-frequencies-tables from "arith_cf_ng [pki=0][545]" to "arith_cf_ng[pki=31][545]" available for the decoding of the group index is 545, as 544 different group indexes and an escape symbol can be decoded. Subsequently, a group index ng may be obtained by executing a function "arith_decode()", taking into consideration the selected cumulative frequencies-table. When deriving the group index ng, bits named "acod_ng" of the bitstream 210 may be evaluated (see Fig. 6g).

The algorithm 312ba also comprises checking whether the group index ng is equal to an escape symbol "ARITH_ESCAPE" or not. If the group index is not equal to the arithmetic escape symbol, the algorithm 312ba is aborted ("break"-condition) and the remaining instructions of the algorithm 312ba are therefore skipped. Accordingly, execution of the process is continued with the element index decoding 312c (if required) or with the most-significant-bitplane-determination 312d. In contrast, if the decoded group index ng is identical to the arithmetic escape symbol "ARITH_ESCAPE", the level value lev is increased by two. Also, if the algorithm 312ba is executed for the first time, i.e. if j=0, the second state value t is increased by 4194304, and the second state value t is set to 0 otherwise. Also, the variable j is set to 1 prior to the repetition of the algorithm 312ba. As

mentioned, the algorithm 312ba is repeated until the decoded group index ng is different from the arithmetic escape symbol.

As soon as the group index decoding 312b is completed, i.e. a group index value different from the arithmetic escape symbol has been decoded, the element index decoding 312c is executed, if required. For this purpose, a cardinal (number of elements) of the group having the group index ng is determined, wherein the cardinal mm of the group designated by the group index ng is described by the eight least-significant bits (bits 0-7) of a table entry "dgroups [ng]" of the table "dgroups" at table position ng. If the cardinal mm of the group designated by the group index ng is larger than one, the element index ne is obtained by executing an algorithm 312ca. Otherwise, the element index ne may optionally be set to 0, or a different default value. For example, the operation ne=0 may be executed before the condition statement "if(mm>1)". The algorithm 312ca comprises the determination of a start address "cum_freq" of an appropriate cumulative-frequencies-table or a cumulative-frequencies-subtable. For example, the variable "cum_freq" may be set to a sum of the start address of the cumulative-frequencies-table "arith_cf_ne" and the value (mm) * (mm-1)/2, as shown in Fig. 3. Also, the variable cfl may be initialized to an appropriate length value of the respective cumulative-frequencies-table or cumulative-frequencies-subtable, which is equal to the number of elements mm within the group of index ng Subsequently, the element index ne may be obtained by performing the function "arith_decode()", wherein the selected cumulative-frequencies-table (e.g. a subtable of the table "arith_cf_ne") associated to the encoding of the element index is used.

Subsequently, the determination 312d of the most-significant bit-plane is executed. For this purpose, an entry of the table "dgvectors" is evaluated, the index of which is determined by the most significant bits (e.g. bits 8-15) of the value j of the table "dgroups" at element index ng and by the element index ne, as can be seen in Fig. 3. To be more specific, the value of the most-significant bit-plane of a first spectral value "a" (of a tuple of spectral values) is determined by an entry of the table "dgvectors" at an element index 4*(j>>8+ne). Similarly, the value of the most-significant bit plane of a second spectral value "b" (of a tuple of spectral values) is obtained by evaluating the entry of the table "dgvectors" at index 4* (j>>8+ne) +1. Similarly, a value of the most-significant plane of a third spectral value "c" and of a fourth spectral value "d" (of the tuple of spectral values) is obtained, as shown in Fig. 3 at reference numeral 312d.

Subsequently, the less-significant bit planes are obtained, for example as shown at reference numerals 312e in Fig. 3. For each less significant bit plane of the tuple, 1 over 16 binary

combinations is decoded. However, it should be noted that the concept for obtaining the values of the less-significant bit planes is not of particular relevance for the present invention.

4.2. Decoding Order (Fig. 4)

In the following, the decoding order of the spectral values will be described.

4-tuples of quantized spectral coefficients are noiselessly coded and transmitted (e.g. in the bitstream) starting from the lowest-frequency coefficient and progressing to the highest-frequency coefficient.

Coefficients from an advanced audio coding (for example obtained using a modified-discrete-cosine-transform, as discussed in ISO/IEC 14496, part3, subpart 4) are stored in an array called "x_ac_quant[g][win][sfb][bin]", and the order of transmission of the noiseless-coding-codeword (e.g. acod_ng, acod_ne, acod_r) is such that when they are decoded in the order received and stored in the array, "bin" (the frequency index) is the most rapidly incrementing index and "g" is the most slowly incrementing index. Within a codeword, the order of decoding is a, b, c, d. In other words, the values a, b, c, d, are spectral values of adjacent frequencies, wherein the spectral value a is associated to a lower frequency than the spectral value b, the spectral value b is associated to a lower frequency than the spectral value c, and the spectral value c is associated to a lower frequency than the spectral value d.

Coefficients from the transform-coded-excitation (tcx) are stored directly in an array x_tcx_invquant[win][bin], and the order of the transmission of the noiseless coding codewords is such that when they are decoded in the order received and stored in the array, "bin" is the most rapidly incrementing index and "win" is the slowest incrementing index. Within a codeword, the order of decoding is a, b, c, d. In other words, if the spectral values describe a transform-coded-excitation of the linear-prediction filter of a speech coder, the spectral values a, b, c, d are associated to adjacent and increasing frequencies of the transform-coded-excitation.

Notably, the audio decoder 200 may be configured to apply the decoded frequency-domain audio representation 232, which is provided by the arithmetic decoder 230, both for a "direct" generation of a time-domain audio signal representation using a frequency-domain to time-domain signal transform and for an "indirect" provision of an audio signal representation using both a frequency-domain to time-domain decoder and a linear-

prediction-filter excited by the output of the frequency-domain to time-domain signal transformer.

In other words, the arithmetic decoder 200, the functionality of which is discussed here in detail, is well-suited for decoding spectral values of a time-frequency-domain representation of an audio content encoded in the frequency-domain and for the provision of a time-frequency-domain representation of a stimulus signal for a linear-prediction-filter adapted to decode a speech signal encoded in the linear-prediction-domain. Thus, the arithmetic decoder is well-suited for use in an audio decoder which is capable of handling both frequency-domain-encoded audio content and linear-predictive-frequency-domain-encoded audio content.

4.3. Context Initialization (Figs. 5a and 5b)

In the following, the context initialization, which is performed in a step 310, will be described. First, the flag "arith_reset_flag", which may be part of the bitstream, determines if the context should be reset. If the flag is TRUE, the function "arith_reset_context()", which is shown in Fig. 5a, is called. Otherwise, if the flag "arith_reset_flag" is FALSE, a mapping is done between the past context and the current context in accordance with the algorithm "arith_map_context()", which is shown in Fig. 5b.

As can be seen, the current context is stored in a global variable q [2][290], which takes the form of an array having a first dimension of 2 and a second dimension of 290. A past context is stored in a variable qs [258], which takes the form of a table having a dimension of 258. The variable "previous_lg/4" describes a number of 4-tuples of a past context.

As can be seen, the reset of the context, which is performed by the function "arith_reset_context()", comprises an initialization of the entries "a", "b", "c", "d" of the arrays q and qs (designated, for example, with qs[i].a , q[0][i].a and q[1][i].a) to zero. In addition, the entries "v" of the arrays q and qs (designated with qs[i].v, q[0][i].v and q[1][i].v) are initialized to –1. Also, the variable "previous_lg" is initialized to 1024.

If, however, it is decided not to reset the context, a mapping of the context may be performed in accordance with the algorithm "arith_map_context()". As can be seen, the mapping is dependent on the core mode, wherein "core_mode==1" indicates that the spectral values to be decoded are associated with a linear-predictive-frequency-domain-encoded audio frame, and wherein "core_mode==0" indicates that the spectral values to be

decoded are associated with a frequency-domain-encoded audio frame. It should be noted here that the function "arith_map_context()" sets the entries q[0][i] of the current context array q to the values qs[i] of the past context array qs, if the number of spectral values associated with the current frequency-domain-encoded audio frame is identical to the number of spectral values associated with the previous audio frame for i=0 to i=lg/4+1.

However, a more complicated mapping is performed if the number of spectral values associated to the current audio frame is different from the number of spectral values associated to the previous audio frame. However, details regarding the mapping in this case are not particularly relevant for the key idea of present invention, such that reference is made to the pseudo program code of Fig. 5b for details.

4.4 State Value Computation (Fig. 5c)

In the following, the state value computation 312a will be described in more detail.

It should be noted that the first state value s (as shown in Fig. 3) can be obtained as a returned value of the function "arith_get_context(i)", a pseudo program code representation of which is shown in Fig. 5c.

Regarding the computation of the state value, reference is also made to Fig. 4, which shows the context used for a state calculation. Fig. 4 shows a two-dimensional representation of tuples of spectral values, both over time and frequency. An abscissa 410 describes the time, and an ordinate 412 describes the frequency. As can be seen in Fig. 4, a tuple 420 to decode is associated with a time index t0 and a frequency index i (keeping in mind that the spectral values of the tuple 420 to decode are associated to four different frequencies). As can be seen, for the time index t0, the tuples having frequency indices i-1 and i-2 are already decoded at the time at which the tuple 420 having the frequency index i is to be decoded. As can be seen from Fig. 4, a tuple 430 having a time index t0 and a frequency index i-1 is already decoded before the tuple 420 is decoded, and the tuple 430 is considered for the context which is used for decoding the tuple 420. Similarly, a tuple 440 having a time index t-1 and a frequency index i-1, a tuple 444 having a time index t-1 and a frequency index i, and a tuple 448 having a time index t-1 and a frequency index i+1 are already decoded before the tuple 420 is decoded, and are considered for the determination of the context which is used for decoding the tuple 420. In contrast, some other tuples already decoded, which are represented by squares having dashed lines, and other tuples, which are not yet decoded and which are shown by circles having dashed lines, are not used for determining the context for decoding the tuple 420.

Taking reference now to Fig. 5c, which shows the functionality of the function "arith_get_context()", some more details regarding the calculation of the first context value "s" will be described.

The function "arith_get_context()" comprises a variable initialization 530a, during which the variables t0, t1, t2 and t3 are initialized in dependence on the entries "v" of the array q at index positions (0, i), (1, i-1), (0, i-1) and (0, i+1). Accordingly, the variables t0 to t3 are initialized with the values of the entries "v", which are associated to the tuples 444, 430, 440, 448 respectively as shown in Fig. 4.

It should also be noted that the function "arith_get_context()" performs a subsequent check of a plurality of conditions, wherein the function "arith_get_context()" is terminated as soon as a "return" instruction is reached, wherein the return instruction (or operator) serves to return its operand (following the return instruction or operator) as the state value s.

The execution of the function "arith_get_context()" comprises a first condition check 530b. If it is found that (the values of) all the variables t0, t1, t2 and t3 are smaller than 10, the return value is computed as shown at reference numeral 530b, and the function "arith_get_context()" is terminated with the return of said return value.

The execution of the function "arith_get_context()" also comprises a second condition check 530c. If it is found in the second condition check 530c that all the variables t0, t1, t2 and t3 are smaller than 34, the variables t2 and t3 are conditionally modified, as shown at the reference numeral 530c, and the return value is computed as shown at reference numeral 530c. In particular, the variable t2 is set to 2 if the variable t2 is greater than 1 and smaller than 10. Otherwise, if the variable t2 is greater than or equal to 10 the variable t2 is set to 3. Similarly, the variable t3 is set to 2, if the variable t3 is greater than 1 and smaller than 10. Otherwise, if the variable t3 is greater than or equal to 10, the variable t3 is set to 3. Accordingly, the range of values of the variable t2 is limited to a maximum positive value of 3.

If, however, neither the condition of the first condition check 530b nor the condition of the second condition check 530c is fulfilled, a third condition check 530d is performed. If it is found in the third condition check 530d that both the variables t0 and t1 are smaller than 90, then the return value is computed as shown in reference numeral 530d, wherein the values of the variables t2 and t3 are left out of consideration.

If, however, none of the conditions of the first condition check 530b, of the second condition check 530c and of the third condition check 530d is fulfilled, a fourth condition check 530e is performed, in which it is determined whether the variables t0 and t1 are both smaller than 544. If this is the case, the return value is computed as shown in reference numeral 530e, and the function "arith_get_context()" is terminated.

If, however, none of the condition checks 530b, 530c, 530d, 530e results in the termination of the function "arith_get_context()", a context computation 530f is performed. The context computation 530f comprises a variable initialization 530fa, a variable rescaling 530fb, a table-based value adaptation 530fc and a return value computation 530fd. In the variable initialization 530fa, variables a0, b0, c0, d0 are set to the values of the entries "a", "b", "c", "d" of the array q at the array position (0,i) if the variable t0 takes a value greater than 1. It corresponds to the values of the 4-tuple 444 of the Fig. 4. Otherwise, if the value of the variable t0 is not greater than 1, the variables a0, b0, c0 and d0 are initialized to 0. Similarly, if the value of the variable t1 is greater than 1, the variables a1, b1, c1, d1 are initialized to the values of the entries "a", "b", "c", "d" of the array q at the position (1, i-1), which corresponds to the values of the 4-tuple 430 of the Fig.4.

Accordingly, the variables a0, b0, c0 and d0 are set to the spectral values a, b, c, d of a previously-decoded tuple of spectral values of time index t-1 and frequency index i, if the value of the variable t0 is greater than 1. Similarly, the variables a1, b1, c1, d1 are set to the spectral values a, b, c, d of the previously-decoded tuple of spectral values and the time index t0 and frequency index i-1.

Subsequently, the variables a0, b0, c0, d0, a1, b1, c1, d1 are iteratively rescaled in that the number representations are iteratively shifted to the right by one bit until all of the variables a0, b0, c0, d0, a1, b1, c1, d1 are in a range between –4 and +3, including the boundaries –4 and +3. After the variable rescaling 530fb, the variable l indicates how often the set of variables a0, b0, c0, d0, a1, b1, c1, d1 has been shifted to the right, wherein at least one shift-to-the-right operation is performed. Accordingly, adapted variables a0, b0, c0, d0, a1, b1, c1, d1 are obtained, which are all in a range between –4 and +3.

Subsequently, the table-based values adaptation 530fc is performed. For this purpose, the variable t0 is set to a value, which is determined by an entry of the table (or array) "egroups", if the variable t0 is greater than 1. As can be seen, the entry at the position (4+a0, 4+b0, 4+c0, 4+d0) is used for this purpose. Similarly, if the value t1 is greater than 1, the variable t1 is set to a value, which is determined by an entry of the table "egroups" at a table position (4+a1, 4+b1, 4+c1, 4+d1).

Finally, a return value is computed in dependence on the variable l (which indicates how often a shift-to-the-right operation has been applied), and also in dependence on the variables t0 and t1, as shown at reference numeral 530fd.

Accordingly, it can generally be said that the return value of the function "arith_get_context" is determined by the most-relevant bit planes (e.g. the three most-relevant bits) of the tuples 444, 430, 440 and 448 of the Fig. 4.

Also, it should be noted that a table lookup is performed if the variable t0 is greater than or equal to 544 or the variable t1 is greater than or equal to 544, while a numerical computation of the return value, using multiplications and additions, is used otherwise. Thus, a more liberate and more detailed computation of the return value of the function arith_get_context is performed if one of the variables t0 and t1 is greater than or equal 544.

It should be noted that in the Fig. 3 of reference numeral 312a, the variable "lev" is derived from the returned value s of the function arith_get_context(i). The variable lev is derived from the value s by shifting the value to the right by 24 bits. The state variable t is also derived from the value s by performing a mask operation of the value s by means of a "and" operation between the value s and the hexadecimal value "0xFFFFFF" and by adding a value of "1" to the result of the end operation.

4.5 Group Index Decoding (Figs. 5d, 5e)

In the following, the process 312b of group index decoding will be discussed, which process 312b is based on a previous calculation of the state value t as described above. Also, the iterative execution of the algorithm 312ba comprises a call of the function "arith_get_pk()" with the state value t (as shown in Fig. 3) as a parameter.

4.5.1. Function "arith_get_pk()" (Fig. 5d)

The function "arith_get_pk()" will subsequently be described in reference to Fig. 5d. The execution of the function "arith_get_pk()" comprises the initialization of an array psci with 28 values, as shown at reference numeral 540a. In addition, the function "arith_get_pk()" comprises the initialization of a pointer p and of the variables i, j, as shown at reference numeral 540b. The algorithm "arith_get_pk()" also comprises an initialization of the variable i to a value, which is equal to 63*t, wherein t is the parameter handed over to the function "arith_get_pk()", when the function "arith_get_pk()" is called. Accordingly, the

input variable s of the function "arith_get_pk()" may be identical to the variable t of the algorithm "tuples_decode()" as shown in Fig. 3. The initialization of the variable i is shown at reference numeral 540c.

The function "arith_get_pk()" also comprises an iterative execution of a hash table access 540d, wherein the hash table access 540d is repeated until a "break" condition is reached, or until a "return" operator is reached. If the "break" condition is reached, a range-based provision 540e of a return value is performed. If, however, the return operator is reached, the operand of the return operator is returned and the function "arith_get_pk()" is terminated.

The hash table access 540d comprises an iterative execution of a first step 540da, a second step 540db, a third step 540dc and a fourth step 540dd. In the first step 540da, the variable j is set to the value of an entry of the table "ari_pk_hash", wherein the index of the entry is determined by the seven least-significant bits of the variable i. In the second step 540db, it is determined whether the value of the variable j, which is obtained in the first step 540da, takes the hexadecimal value of 0xFFFFFFFF. If this is the case, the iterative execution of the hash table access 540d is aborted and the execution of the algorithm "arith_get_pk()" is continued with the range-based provision of a return value. In other words, if the entry of the table "ari_pk_hash()", which is addressed by the seven least-significant bits of the variable i, takes the escape value of 0xFFFFFFFF, it is assumed that the state defined by the input variable t of the function "arith_get_pk()" is a so-called "non-significant" state, to which a return value should be assigned using the range-based provision 540e of the return value. In the third step 540dc of the hash table access 540d, it is checked whether the most significant bits (e.g. bits 8 to 31) of the value of the variable j are equal to the value of the input variable t of the function "arith_get_pk()". If this is the case, the eight least significant bits (bits 0 to 7) of the variable j are returned as a return value of the function "arith_get_pk()", and the function "arith_get_pk()" is terminated. If, however, it is found that the condition of the third step 540dc is not fulfilled, the variable i is incremented by 1 (step 540dd) and the hash table access 540d is repeated, starting from the first step 540da.

The range-based provision 540e of a return value comprises an initialization 540ea of the pointer p to a starting point within the array psci. The starting point is determined by bits 23 and 24 of the input variable t of the function "arith_get_pk()", which corresponds to the number of escape symbol "ARITH_ESCAPE" already decoded for the present tuple to decode. The pointer p is initialized to point at the first entry (24) of the array psci if the bits 23 and 24 of the input variable t take the values "00", to the eighth entry (30) of the array psci, if the bits 23 and 24 of the input variable s take the values "01", to the 15th entry (5)

of the array psci, if the 23rd and 24th bit of the input variable s take the values "10" and to the 22nd entry (5) of the array psci, if the 23rd and 24th bit of the input variable t take the values "11". In a subsequent step 540eb, the variable j is set to take the value represented by the 22 least-significant bits (bits 1 to bit 22) of the input variable t, as shown at reference numeral 540eb. Subsequently, a decision is made which entry of the array psci is returned as the return value of the algorithm "arith_get_pk()". The following decision is made:

- If the value of j is smaller than 436961, and if the value of j is also smaller than 252001, and if the value of j is also smaller than 243001, the entry at the starting point determined in the step 540ea is returned;
- If the value of j is smaller than 436961, and the value of j is also smaller than 252001 and if the value of j is not smaller than 243001, the first entry after the starting point determining the step 540ea is returned;
- If the value of j is smaller than 436961, and if the value of j is not smaller than 252001, and if the value of j is smaller than 288993, the second entry after the starting point determined in step 540ea is returned;
- If the value of j is smaller than 436961, and if the value of j is not smaller than 252001, and if the value of j is not smaller than 288993, the third entry after the starting point determined in the step 540ea is returned;
- If the value of j is not smaller than 436961, and if the value of j is smaller than 1609865 and if the value of j is also smaller than 880865, the fourth entry after the starting point determined in the step 540ea is returned;
- If the value of j is not smaller than 436961, and if the value of j is smaller than 1609865, and if the value of j is not smaller than 880865, the fifth entry after the starting point determined in the step 540ea is returned; and
- If the value of j is not smaller than 436961, and if the value of j is also not smaller than 1609865, the sixth value after the starting point determined in the step 540ea is returned.

For more details, reference is made to the algorithm at reference numeral 540ec in Fig. 5.

To summarize the above, the function "arith_get_pk()", which is called with the state value t, provides the value pki as a return value, as can be seen in Fig. 3 at reference numeral 312ba.

The value of the variable pki is used to select a cumulative-frequencies-table for the execution of the function "arith_decode()" as it has been discussed with reference to Fig.3. Accordingly, the variable "cum_freq[]" is initialized appropriately to designate the selected cumulative-frequencies-table.

4.5.2. Function "arith_decode()" (Fig. 5e)

In the following, the functionality of the function "arith_decode()" will be discussed in detail taking reference to Fig. 5e. It should be noted that the function "arith_decode()" uses the helper function "arith_first_symbol (void)", which returns TRUE, if it is the first symbol of the sequence and FALSE otherwise. The function "arith_decode()" also uses the helper function "arith_get_next_bit()", which gets and provides the next bit of the bitstream.

In addition, the function "arith_decode()" uses the global variables "low", "high" and "value". Further, the function "arith_decode()" receives, as an input variable, the variable "cum_freq[]", which points towards a first entry or element (having element index or entry index 0) of the selected cumulative-frequencies-table. Also, the function "arith_decode()" uses the input variable cfl, which indicates the length of the selected cumulative-frequencies-table designated by the variable "cum_freq[]".

The function "arith_decode()" comprises, as a first step, a variable initialization 550a, which is performed if the helper function "arith_first_symbol()" indicates that the first symbol of a sequence of symbols is being decoded. The value initialization 550a initializes the variable "value" in dependence on a plurality of, for example, 20 bits, which are obtained from the bitstream using the helper function "arith_get_next_bit", such that the variable "value" takes the value represented by said bits. Also, the variable "low" is initialized to take the value of 0, and the variable "high" is initialized to take the value of 1048575.

In a second step 550b, the variable "range" is set to a value, which is larger, by 1, than the difference between the values of the variables "high" and "low". The variable "cum" is set to a value which represents a relative position of the value of the variable "value" between the value of the variable "low" and the value of the variable "high". Accordingly, the variable "cum" takes, for example, a value between 0 and 2^{16} in dependence on the value of the variable "value".

The pointer p is initialized to a value which is smaller, by 1, than the starting address of the selected cumulative-frequencies-table.

The algorithm "arith_decode()" also comprises an iterative cumulative-frequencies-table-search 550c. The iterative cumulative-frequencies-table-search is repeated until the variable cfl is smaller than or equal to 1. In the iterative cumulative-frequencies-table-search 550c, the pointer variable q is set to a value, which is equal to the sum of the current value of the pointer variable p and half the value of the variable cfl. If the value of the entry *q of the selected cumulative-frequencies-table, which entry is addressed by the pointer variable q, is larger than the value of the variable "cum", the pointer variable p is set to the value of the pointer variable q, and the variable cfl is incremented. Finally, the variable cfl is shifted to the right by one bit, thereby effectively dividing the value of the variable cfl by 2 and neglecting the modulo portion.

Accordingly, the iterative cumulative-frequencies-table-search 550C effectively compares the value of the variable "cum" with a plurality of entries of the selected cumulative-frequencies-table, in order to identify an interval within the selected cumulative-frequencies-table, which is bounded by entries of the cumulative-frequencies-table, such that the value cum lies within the identified interval. Accordingly, the entries of the selected cumulative-frequencies-table define intervals, wherein a respective symbol value is associated to each of the intervals of the selected cumulative-frequencies-table. Also, the widths of the intervals between two adjacent values of the cumulative-frequencies-table define probabilities of the symbols associated with said intervals, such that the selected cumulative-frequencies-table in its entirety defines a probability distribution of the different symbols (or symbol values). Details regarding the available cumulative-frequencies-tables will be discussed below taking reference to Fig. 16.

Taking reference again to Fig. 5e, the symbol value is derived from the value of the pointer variable p, wherein the symbol value is derived as shown at reference numeral 550d. Thus, the difference between the value of the pointer variable p and the starting address "cum_freq" is evaluated in order to obtain the symbol value, which is represented by the variable "symbol".

The algorithm "arith_decode" also comprises an adaptation 550e of the variables "high" and "low". If the symbol value represented by the variable "symbol" is different from 0, the variable "high" is updated, as shown at reference numeral 550d. Also, the value of the variable "low" is updated, as shown at reference numeral 550e. The variable "high" is set to a value which is determined by the value of the variable "low", the variable "range" and

the entry having the index "symbol –1" of the selected cumulative-frequencies-table. The variable "low" is increased, wherein the magnitude of the increase is determined by the variable "range" and the entry of the selected cumulative-frequencies-table having the index "symbol". Accordingly, the difference between the values of the variables "low" and "high" is adjusted in dependence on the numeric difference between two adjacent entries of the selected cumulative-frequencies-table.

Accordingly, if a symbol value having a low probability is detected, the interval between the values of the variables "low" and "high" is reduced to a narrow width. In contrast, if the detected symbol value comprises a relatively large probability, the width of the interval between the values of the variables "low" and "high" is set to a comparatively large value. Again, the width of the interval between the values of the variable "low" and "high" is dependent on the detected symbol and the corresponding entries of the cumulative-frequencies-table.

The algorithm "arith_decode()" also comprises an interval renormalization 550f, in which the interval determined in the step 550e is iteratively shifted and scaled until the "break"-condition is reached. In the interval renormalization 550f, a selective shift-downward operation 550fa is performed. If the variable "high" is smaller than 524286, nothing is done, and the interval renormalization continues with an interval-size-increase operation 560fb. If, however, the variable "high" is not smaller than 524286 and the variable "low" is greater than or equal to 524286, the variables "values", "low" and "high" are all reduced by 524286, such that an interval defined by the variables "low" and "high" is shifted downwards, and such that the value of the variable "value" is also shifted downwards. If, however, it is found that the value of the variable "high" is not smaller than 524286, and that the variable "low" is not greater than or equal to 524286, and that the variable "low" is greater than or equal to 262143 and that the variable "high" is smaller than 786429, the variables "value", "low" and "high" are all reduced by 262143, thereby shifting down the interval between the values of the variables "high" and "low" and also the value of the variable "value". If, however, neither of the above conditions is fulfilled, the interval renormalization is aborted.

If, however, any of the above-mentioned conditions, which are evaluated in the step 550fa, is fulfilled, the interval-increase-operation 550fb is executed. In the interval-increase-operation 550fb, the value of the variable "low" is doubled. Also, the value of the variable "high" is doubled, and the result of the doubling is increased by 1. Also, the value of the variable "value" is doubled (shifted to the left by one bit), and a bit of the bitstream, which is obtained by the helper function "arith_get_next_bit" is used as the least-significant bit.

Accordingly, the size of the interval between the values of the variables "low" and "high" is approximately doubled, and the precision of the variable "value" is increased by using a new bit of the bitstream. As mentioned above, the steps 550fa and 550fb are repeated until the "break" condition is reached, i.e. until the interval between the values of the variables "low" and "high" is large enough.

Regarding the functionality of the algorithm "arith_decode()", it should be noted that the interval between the values of the variables "low" and "high" is reduced in the step 550e in dependence on two adjacent entries of the cumulative-frequencies-table referenced by the variable "cum_freq". If an interval between two adjacent values of the selected cumulative-frequencies-table is small, i.e. if the adjacent values are comparatively close together, the interval between the values of the variables "low" and "high", which is obtained in the step 550e, will be comparatively small. In contrast, if two adjacent entries of the cumulative-frequencies-table are spaced further, the interval between the values of the variables "low" and "high", which is obtained in the step 550e, will be comparatively large.

Consequently, if the interval between the values of the variables "low" and "high", which is obtained in the step 550e, is comparatively small, a large number of interval renormalization steps will be executed to re-scale the interval to a "sufficient" size (such that neither of the conditions of the condition evaluation 550fa is fulfilled). Accordingly, a comparatively large number of bits from the bitstream will be used in order to increase the precision of the variable "value". If, in contrast, the interval size obtained in the step 550e is comparatively large, only a smaller number of repetitions of the interval normalization steps 550fa and 550fb will be required in order to renormalize the interval between the values of the variables "low" and "high" to a "sufficient" size. Accordingly, only a comparatively small number of bits from the bitstream will be used to increase the precision of the variable "value" and to prepare a decoding of a next symbol.

To summarize the above, if a symbol is decoded, which comprises a comparatively high probability, and to which a large interval is associated by the entries of the selected cumulative-frequencies-table, only a comparatively small number of bits will be read from the bitstream in order to allow for the decoding of a subsequent symbol. In contrast, if a symbol is decoded, which comprises a comparatively small probability and to which a small interval is associated by the entries of the selected cumulative-frequencies-table, a comparatively large number of bits will be taken from the bitstream in order to prepare a decoding of the next symbol.

Accordingly, the entries of the cumulative-frequencies-tables reflect the probabilities of the different symbols and also reflect a number of bits required for decoding a sequence of symbols. By varying the cumulative-frequencies-table in dependence on a context, i.e. in dependence on previously decoded symbols, for example by selecting different cumulative-frequencies-tables in dependence on the context, stochastic dependencies between the different symbols can be exploited, which allows for a particular bitrate-efficient encoding of the subsequent (or adjacent) symbols.

To summarize the above, the function "arith_decode()", which has been described with reference to Fig. 5e, is called with the cumulative-frequencies-table "arith_cf_ng[pki][]", corresponding to the index pki returned by the function "arith_get_pk()" to determine the group index ng.

4.5.3. Escape Mechanism

While the decoded group index ng is the escape symbol "ARITH_ESCAPE", an additional group index ng is decoded and the variable lev is incremented by 2. Accordingly, an information is obtained about the numeric significance of the most-significant bit-plane and also about a number of less-significant bit-planes to be decoded. If an escape symbol "ARITH_ESCAPE" is decoded for the first time for the present tuple to decode, the state variable t is then incremented by the value 4194304, which corresponds to set to "1" the 23th bit of the variable t. If an escape symbol is decoded for the second and more time, the state variable t is then set to zero. In both cases, when a escape symbol "ARITH_ESCAPE" is decoded, the updated state variable t is then used for a new iteration of the group index decoding 312b.

4.6.Element Index Decoding (Fig. 5f)

Once the decoded group index is not the escape symbol, "ARITH_ESCAPE", the number of elements, mm, within the group ng and the group offset, og are deduced by looking up the table "dgroups[]" in accordance with the algorithm shown in Fig. 5f.

In other words, the variable mm is set to a value, which is determined by the least-significant bits (e.g. bits 0-7) of the entries of the table "dgroups[]" at a table position determined by the group index ng. Similarly, the group offset og is determined by the more

significant bits (bit 8 and onwards) of the entry of the table "dgroups[]", which entry is determined by the position offset defined by the variable ng.

If the variable mm is greater than 1, i.e. if the group determined by the group index ng comprises more than one element, the element index ne is decoded by calling the function "arith_decode()" with the cumulative-frequencies-table

arith_cf_ne + ((mm*(mm-1))>>1)[] and with the length of the cumulative-frequencies-table equals to mm.

In other words, a subsection of the table "arith_cf_ne[]" is selected, wherein the cumulative-frequencies-table "arith_cf_ne[]" in its entirety describes probability distributions for a plurality of different numbers of elements of a group selected by the group index ng. It should be noted that offsets of the different subsections (or subtables) of the cumulative-frequencies-table "arith_cf_ne[]" are described by the formula (mm*(mm-1))>>1.

It should be noted that preferably the variable "cum_freq", which is used as an input variable of the algorithm "arith_decode()", is initialized to the starting address of the subsection (or subtable) of the table or array "arith_cf_ne[]" which is associated to the number of elements mm of the current group described by the group index ng. Also, the variable cfl, which is an input variable of the algorithm "arith_decode()", is initialized to the value mm. Subsequently, the function "arith_decode()" is called, the operation of which has been described in detail above. However, for the decoding of the element index, the function "arith_decode()" uses a subtable of the table or array "arith_cf_ne[]", rather than one of the cumulative-frequencies tables "arith_cf_ng[pki=0][545]" to "arith_cf_ng[pki=31][545]". Consequently, the element index ne is provided as a return value of the function "arith_decode()", which uses a number of bits of the bitstream to obtain the element index ne.

4.7. Most-Significant Bit-Plane Determination (Fig. 5g)

Once the element index ne is decoded and returned as a return value of the function "arith_decode()", the most-significant signed 2-bits wise plane of the 4-tuple can be derived using the table "dgvector[]" in accordance with the algorithm shown in Fig. 5g.

For example, a first spectral value "a" of the tuple of spectral values can be set to an entry of the table or array "dgvectors[]", wherein the array element index (or table element

index, or briefly "element index" or "entry index") is determined as 4*(og+ne). Similarly, the second spectral value "b" of the tuple of spectral values can be set to an entry of the array "dgvectors[]", wherein the array element index is determined by 4*(og+ne)+1. A third spectral value "c" of the tuple of spectral values can be set to an entry of the array "dgvectors[]", wherein an element index is determined by 4*(og+ne)+2. A fourth spectral value "d" of the tuple of spectral values can be set to an entry of the array "dgvectors[]", wherein the element index is determined by 4*(og+ne)+3. Thus, the spectral values "a"," b", "c", "d", which represent the most-significant bit planes of the tuple of spectral values, are derived from the array "dgvectors[]", wherein the entries of the array determining the spectral values "a", "b", "c", "d" are selected in accordance with the group index ng and the element index ne (if available).

4.8. Less-Significant Bit-Plane Determination (Fig. 5h)

The remaining bit planes are then decoded from the most-significant to the lowest significant level by calling lev times the function "arith_decode()" with the cumulative-frequencies-table "arith_cf_r[]". For this purpose, the input variable "cum_freq" of the function "arith_decode()" may be initialized to the starting address of the array "arith_cf_r[]". Also, the input variable cfl of the function "arith_decode()" may be initialized to an appropriate value representing the length of the table "arith_cf_r[]", which is in case of a tuple of dimension 4 equals to 16.

The function "arith_decode()" can return the variable r, which represents the binary values of one of the less-significant bit-plane of the decoded tuple. The decoded bit-plane r permits to refine the decoded 4-tuple according to the algorithm shown in Fig. 5h.

In other words, when "adding" a less-significant bit plane, the first spectral value "a" is multiplied by 2 (or, equivalently, shifted to the left by one bit), and the least-significant bit (bit 0) of the value r is added as the new least-significant bit (which may be done using an OR operation). The second spectral value "b" is multiplied by 2, and the second bit (bit 1) of the value r is added as a least-significant value of the spectral value "b". The third spectral value "c" is multiplied by 2 (or, equivalently, shifted to the left by one bit), and the third bit (bit 2) of the value r is added as the least-significant bit. The fourth spectral value d" is multiplied by 2 (or, equivalently, shifted to the left by one bit), and the fourth bit (bit 3) of the value r is added as a least-significant bit. For details, reference is made to the algorithm shown in Fig. 5h.

4.9. Context Update (Fig. 5i)

Once the 4-tuple (a, b, c, d) is completely decoded, i.e. all of the less-significant bit planes have been added, the context tables q and qs are updated by calling the function "arith_update_context()". In the following, details regarding the function "arith_update_context()" will be described taking reference to Fig. 5i, which shows a pseudo-program code representation of said function.

The function "arith_update_context" receives, as input variables, the spectral values "a", "b", "c", "d" of the decoded 4-tuple, the index "i" of the 4-tuple to decode (or the decoded 4-tuple) and the number lg/4 of 4-tuples associated with the current audio frame.

The function "arith_update_context()" comprises a step 580a of copying the spectral values "a", "b", "c", "d", into the array q. For example, the entry "a" of the array q at the position (1, i) (also designated with "q[1][i].a") is set to take the first spectral value "a". The entry "b" of the array q at the position (1, i) is set to the second spectral value "b". The entry "c" of the array q at the position (1, i) is set to the third spectral value "c", and the entry "d" of the array q at the position (1, i) is set to the fourth spectral value "d". Accordingly, the spectral values "a", "b", "c", "d" are stored in the entries "a", "b", "c", "d" of the array q at the position (1, i).

The function "arith_update_context()" also comprises a step 580b of setting the entry "v" of the array q at the position (1, i). If one of the spectral values "a", "b", "c", "d" of the currently-decoded tuple of spectral values is smaller than –4 or greater than or equal to 4, the entry "'v" of the array q at the position (1, i) is set to the value of 1024. Otherwise, i.e. if all the spectral values "a", "b", "c", "d" are within the range between –4 and +3, including the boundaries, the entry "v" of the array q at the position (1, i) is set to an entry of the table or array "egroups[]" at the position (4+a, 4+b, 4+c, 4+d). Accordingly, the entry "v" of the array q is typically set to a standard value of 1024, if one of the spectral values "a", "b", "c", "d" is comparatively large, thereby causing the function "arith_get_context()" to perform the process by 530f during the decoding of an adjacent tuple of spectral values.

The function "arith_update_context()" also comprises a first mapping 580c, which is performed if the last tuple of spectral values of a current audio frame is decoded and the core mode is the linear-predictive-frequency-domain core mode (for the case of an audio coder switchable between a frequency-domain core mode and a linear-predictive-frequency-domain core mode. The function "arith_update_context()" also comprises a

second mapping 580d, which is performed if the last tuple of spectral values of the current audio frame is decoded and if the core mode is the frequency-domain core mode.

Regarding the first mapping 580c, reference is made to the pseudo-program code of Fig. 5i. Regarding the second mapping 580d, reference is also made to Fig. 5i.

4.10 Summary of the Decoding Process

In the following, the decoding process will be briefly summarized.

4-tuples of quantized spectral coefficients are noiselessly coded and transmitted starting from the lowest-frequency coefficient and progressing to the highest-frequency coefficient. The coefficients from the advanced-audio-coding are stored in an array "x_ac_quant[g][win][sfb][bin]" and the order of transmission of the noiseless coding codewords is such that when they are decoded in the order received and stored in the array, "bin" is the most rapidly incrementing index and "g" is the most slowly incrementing index. Within a codeword, the order of decoding is a, b, c, d.

Coefficients from the transform-coded-excitation are stored directly in an array "x_tcx_invquant[win][bin]", and the order of the transmission of the noiseless coding codewords is such that when they are decoded in the order received and stored in the array, "bin" is the most rapidly incrementing index and "win" is the most slowly incrementing index. Within a codeword, the order of decoding is a, b, c, d.

First, the flag "arith_reset_flag" determines if the context must be reset. If the flag is TRUE, the function "arith_reset_context", a pseudo-program-code representation of which is shown in Fig. 5a, is called. Otherwise, when the flag "arith_reset_flag" is FALSE, a mapping is done between the past context and the current context in accordance with the function "arith_map_context()", a pseudo-program-code representation of which is shown in Fig. 5b.

The noiseless decoder outputs 4-tuples of signed quantized spectral coefficients. At first, the state of the context is calculated based on the four previously decoded groups surrounding the 4-tuple to decode. The state of the context is given by the function "arith_get_context()", a pseudo-program-code representation of which is shown in Fig. 5c.

Once the state is known, the group to which belongs the most-significant signed 2-bits wise plane of the 4-tuple is decoded using the function "arith_decode()", fed with the appropriated cumulative-frequencies-table corresponding to the context state.

The correspondence is made by the function "arith_get_pk()", a pseudo-program-code representation of which is shown in Fig. 5d.

Then, to determine the group index ng, the function "arith_decode()" is called with the cumulative-frequencies-table, "arith_cf_ng[pki][], corresponding to the index returned by the function "arith_get_pk()". The arithmetic coder (or decoder) is an integer implementation using the method of tag generation with scaling. For details, reference is made, for example, to the book "Introduction to Data Compression" of K. Sayood, third edition, 2006, Elsevier Inc. The pseudo-C-code of Fig. 5e describes the used algorithm of the function "arith_decode()".

While the decoded group index ng is the escape symbol, ARITH_ESCAPE, an additional group index ng is decoded and the variable lev is incremented by 2. The state of the context is also adjusted. Once the decoded group index is not the escape symbol, ARITH_ESCAPE, the number of elements, mm, within the group and the group offset, og, are deduced by looking up to the table dgroups[], in accordance with the algorithm shown in Fig. 5f.

The element index ne is then decoded by calling the function "arith_decode()" with the cumulative-frequencies-table (arith_cf_ne+((mm*(mm-1)) >>1)[]. Once the element index is decoded, the most-significant 2-bits wise plane of the 4-tuple can be derived with the table "dgvector[]" in accordance with the algorithm shown in Fig. 5g.

The remaining bit planes are then decoded from the most-significant to the lowest-significant level by calling lev times the function "arith_decode()" with the cumulative-frequencies-table "arith_cf_r[]".

The decoded bit plane r permits to refine the decoded 4-tuple by the algorithm shown in Fig. 5h.

Once the 4-tuple (a, b, c, d) is completely decoded, the context tables q and qs are updated by calling the function "arith_update_context()", which is shown in Fig. 5i.

A legend of definitions is shown in Fig. 5j.

5. Mapping tables

In an embodiment according to the invention, particularly advantageous tables "arith_cf_ng_hash" and "arith_cf_ng" are used for the execution of the function "arith_get_pk", which has been discussed with reference to Fig. 5d, and for the execution of the function "arith_decode", which has been discussed with reference to Fig. 5e.

5.1. Table "arith_cf_ng_hash[]"

A content of a particularly advantageous implementation of the table "arith_cf_ng_hash[]", which takes the role of the table "arith_pk_hash[]", is shown in the table of Fig. 14. It should be noted here that the table of Fig. 14 lists the entries of the table "arith_cf_ng_hash[]. Said entries are referenced by a one-dimensional integer-type entry index (also designated as "element index" or "array index"), which is, for example, designated with "i". As can be seen, a first column 1410, which is an index column, describes starting indices associated to the respective lines 1412a to 1412p. A first value column 1420 shows the values of entries of the table "arith_cf_ng_hash[]" for entry indices identical to the start index shown in the index column 1310. A second value column 1422 shows entries of the table "arith_cf_ng_hash" for entry indices which are large, by one, compared to the start indices shown in column 1410 of the respective line. A third value column 1424 describes entries of the table "arith_cf_ng_hash", for which the element index is larger, by two, than the starting index shown in column 1410 of the corresponding line. Similarly, columns 1426, 1428, 1430, 1432, 1434 show entries of the table "arith_cf_ng_hash", for which the element index is larger, by 3 (column 1526) , by 4 (column 1428), by 5 (column 1430), by 6 (column 1432) or by 7 (column 1434) than the starting index shown in column 1410 of the respective line 1412a to 1412p.

For example, the line 1412a shows, in the columns 1420 to 1434, entries of the table "arith_cf_ng_hash" having element indices 0, 1, 2, 3, 4, 5, 6 and 7. Similarly, the line 1412b shows, in the columns 1420 to 1434, a sequence of entries of the table "arith_cf_ng_hash" having element indices of 8, 9, 10, 11, 12, 13, 14 and 15. For the other lines 1412c to 1412p, the arrangement of the entries discussed above applies analogously.

5.2. Table "arith_cf_ne"

Figs. 15 (1) to 15 (10) show table presentations of entries of the table "arith_cf_ne", which is evaluated by the function "arith_decode()" when decoding the element index ne.

As can be seen, the entries of the table "arith_cf_ne[]" comprise indices between 0 and 2699. A starting index column 1510 describes starting indices associated to the respective lines. A first line is designated, for example, with 1512a, and a second line is designated, for example, with 1512b.

A first entry column 1520 shows entries of the table "arith_cf_ne[]", the entry index of which is determined by the value given in the starting index column 1510 of the corresponding lines (e.g. lines 1512, 1512b). Entry columns 1522, 1524, 1526, 1528, 1530, 1532, 1534 show entries of the table "arith_cf_ne[]" having element indices which are larger, by one (column 1522), by 2 (column1524), by 3 (column 1526), by 4 (column 1528), by 5 (column 1530), by 6 (column 1532) or by 7 (column 1534) than the starting index shown in the starting index column 1510 of the respective line (e.g. line 1512a or 1512b). Accordingly, the first line 1512a shows, in columns 1520 to 1534, elements of the table "arith_cf_ne[]" having entry indices between 0 and 7. The second line 1512b shows, in columns 1520 to 1534, entries of the table "arith_cf_ne[]" having entry indices between 8 and 15.The above discussed rules also apply to the additional lines of the table "arith_cf_ne[]".

5.3. Table "arith_cf_ng[pki][545]

Figs. 16(1) to 16(32) show a set of 32 cumulative-frequencies-tables "arith_cf_ng[pki][545], one of which is selected by an audio encoder 100 or audio decoder 200, for example, for the execution of the function "arith_decode()", i.e. for the decoding of the group index ng. The selected one of the 32 cumulative-frequencies-tables shown in Fig. 16(1) to 16(32) takes the function of the table "cum_freq[]" in the execution of the function "arith_decode()".

As can be seen in Figs. 16(1) to Fig. 16(32), different values of the variable pki are associated to the different tables. A cumulative-frequencies-table index pki=0 is associated to a table 1601, a cumulative-frequencies-table index pki=1 is associated to table 1602, and cumulative-frequencies-table indices pki=2, pki=3, ... pki=31 are associated with tables 1603 to 1632.

The structure of the tables 1601 to 1632 associated with the table indices pki=0 to pki=32 is identical, such that only the structure of the representation of the table 1601 associated

with the table index pki=0 will be discussed in detail. The table 1601 comprises an index column 1640, which shows starting indices associated with respective lines of the table 1601. A first line is designated to 1642a, and a second line is designated with 1642b.

A first entry column 1650 represents entries of the table 1601, the entry indices of which are identical to the starting indices shown in the index column 1640 of the respective lines. Similarly, entry columns 1651 to 1665 show entries of the table 1601 (associated with the table index pki=0) having entry indices which are larger, by 1 (col. 1651), by 2 (col. 1652), by 3 (1653), by 4 (col. 1654), by 5 (col. 1655), by 6 (col. 1656), by 7 (col. 1657), by 8 (col. 1658), by 9 (col. 1659), by 10 (col. 1660), by 11 (col. 1661), by 12 (col. 1662), by 13 (1663), by 14 (col. 1664) or by 15 (col. 1665) than the starting indices, shown in the index column 1640 of the respective lines. Accordingly, the first line 1632a shows the values of the entries of the table 1601 having element indices between 0 (value 16684) and 15 (value 6352). The second line 1642b shows values of entries of the table 1601 having element indices between 16 (value 6202) and 31 (value 3547).

Naturally, the above rules apply to the other lines as well. Also, the arrangement of the element entries in the tables 16(2) to 16(32) is identical to the arrangement of the entries of the table 1601.

5.4. Table "dgroups[]"

Figs. 17(1) and 17(2) show a representation of the entries of a table "dgroups[]", which may be applied by the audio encoder 100 and the audio encoder 200. For example, the table "dgroups[]" may be applied in the execution of the algorithm "tuples_decode()", which is shown in Fig. 3. Also, the table "dgroups[]" may be applied by the algorithm shown in Fig. 5f in order to determine the number mm of elements in a group and the group offset og of a group designated by the group index ng.

The representation of the table "dgroups[]" comprises an index column 1710, which shows start indices associated with respective lines, e.g. a first line 1712a and a second line 1712b of the table representation. A first entry column 1720 of the representation shows entries of the table "dgroups[]", the element indices of which are identical to the starting indices shown in the index column 1710 of the respective lines. Similarly, entry columns 1722, 1724, 1726, 1728, 1730, 1732, 1734 show entries of the table "dgroups[]", the element indices which are larger by 1 (col. 1722), by 2 (col. 1724), by 3 (col. 1726), by 4 (col. 1728), by 5 (col. 1730), by 6 (col. 1732) or by 7 (col. 1734) than the starting indices shown

in the index column 1710 of the respective lines. For example, the first line 1712a shows, in entry columns 1722 to 1734, entries of the table "dgroups[]" having element indices between 0 (col. 1720) and 7 (col. 1734). Similarly, the second line 1712b shows, in columns 1720 to 1734, entries of the table "dgroups[]" having element indices between 8 (col. 1720) and 15 (col. 1734). It should be noted that the entries of the table "dgroups[]" are shown in a hexadecimal notation in Figs. 17 (1) and 17 (2), which is indicated by the prefix "0x". The most-significant hexadecimal digit is shown on the left side and the least-significant hexadecimal digit is shown on the right side.

5.5. Table "dgvectors[]"

Figs. 18(1) to 18(11) show a table representation of the entries of the table "dgvectors[]". The table dgvectors[] may for example be used in the audio encoder 100 or the audio decoder 200. For example, the table "dgvectors[]" may be used in the step 312d of the function "tuples_decode()" shown in Fig. 3 or in the execution of the algorithm of Fig. 5g. Accordingly, the table "dgvectors[]" may be used to map a group index and an element index onto values of a most-significant bit-plane of a tuple of spectral values.

The table representation of Figs. 18(1) to 18(11) comprises an index column 1810, which comprises starting indices associated with the lines (for example a first line 1812a or a second line 1812b) of the table representation. Entry columns 1820 to 1882 show entries of the table "dgvectors[]". For example, the entry column 1820 shows entries of the table "dgvectors[]", entry indices of which are identical to a corresponding starting index shown in the index column 1810 of the respective lines. Subsequent entry columns 1822 to 1882 show, in ascending order, entries of the table "dgvectors[]" which are larger, by 1 (col. 1822), by 2 (col. 1824), by 3 (col. 1826), by 4, by 5, by 6, by 7, by 8, by 9, by 10, by 11, ...by 29 (col. 1878), by 30 (col. 1880) or by 31 (col. 1882) than the starting index values shown in the index column 1810 of the respective lines. Accordingly, the first line 1812a shows, in columns 1820 to 1882, entries of the table "dgvectors[]" having element indices between 0 and 31, wherein the element indices associated with the entries increase monotonically from the left to the right. Similarly, the second line 1812b shows, in columns 1820 to 1882, entries of the table "dgvectors[]" having element indices between 32 and 63 (with the element indices increasing from the left to the right).

5.6. Table "egroups"

Figs. 19(1) to 19(32) show a table representation of a table "egroups[a][d][c][d]", which may also be considered as a 4-dimensional array having four element indices a, b, c, d. It should be noted that each of the element indices a, b, c, d may take values between 0 and 7. The table or array "egroups[a][b][c][d]" may be used in the audio encoder 100 or in the audio decoder 200. For example, the table or array "egroups[a][b][c][d]" may be used in the function "arith_get_context()" to derive the return value and in the function "arith_update_context()" to determine the entry "v" of the array q at entry index (1, 1+i).

The entries of the array "egroups" are shown in 64 tables 1901 to 1964. Different combinations of index values a and b are associated with each of the tables 1901 to 1964. For example, the combination a=0 and b=0 is associated with the first table 1901, and the combination a=0, b=1 is associated to the second table 1902.

It should be noted here that the structure of different tables 1901 to 1964 is identical, such that only the structure of the first table 1901 will be discussed here. The table 1901 comprises an index column 1970 representing values of the third index c associated with the respective lines 1972a to 1972h. Similarly, an index line 1980 represents index values of the fourth index d associated with respective columns 1982a to 1982h of the table 1901. Accordingly, the indices a and b associated to an entry of the tables 1901 to 1964 are written next to the respective table. The index c associated with an entry of one of the tables 1901 to 1964 is determined by the value of the index column of the respective line, and the fourth index d of an entry is determined by the value of the index line of the respective column of the entry. For example, the line 1972 of the table 1901 represents entries of the array "egroups[a][b][c][d]" for the indices a=0, b=0, c=0 and d=(0 to 7) (from the left to the right). Also, the column 1982a represents entries of the array "egroups[a][b][c][d]" with indices a=0, b=0, c=(0 to 7) and d=0, from top to the bottom. Also, it should be noted that the entries of the array "egroups" are represented in hexadecimal notation, which is indicated by the prefix "0x".

6. Performance Evaluation and Advantages

The embodiments according to the invention use an updated set of tables, as discussed above, which reduces significantly the memory requirements for the spectral noiseless coding when compared to a previously-used set of tables. A lossless transcoding is possible within the bitrate constraints.

In the following, the modification of a previously-used noiseless coding underlying the inventive concept will be discussed.

Audio coding concepts, like, for example, the so-called unified-speech-and-audio-coding (USAC) use a context-adapted arithmetic coder (and decoder) for noiselessly (or losslessly) coding the quantized spectral coefficients (for example the spectral coefficients 252). The context adaptation associated with an arithmetic coder (encoder or decoder) permits to achieve high noiseless coding performances. The main drawback of this technology comes from its relatively high complexity in terms of a memory requirement. Indeed, the context adaptation requires a significantly large set of tables modeling different probability distributions. The read-only-memory (ROM) consumption of the previously-used unified-speech-and-audio-coding (USAC) is evaluated to be about 150 kWords, wherein the entropy coder represents around 73 percent of the total requirement.

One of the aims of the present contribution is to propose a new set of tables for the arithmetic coder (or decoder) requiring significantly less memory space while maintaining the original performance of the noiseless coder (encoder or decoder).

In the following, the current status of the previously-implemented unified-speech-and-audio-coding will be described. Fig. 7 shows a table listing the detailed memory requirements of a previously-implemented USAC noiseless coder (encoder or decoder).

It is easily observable from the table of Fig. 7 that the most demanding tables are, by far, the tables "arith_cf_ng_hash[]" and "arith_cf_ng[][]" related to the context-adaptation of the group symbol coding. It is worth notifying that the tables "arith_cf_ne[]" and "arith_cf_r[]" gathering the cumulative frequencies of the element index symbols and the remaining bit plane symbols can easily be algebraically recovered and do not necessarily need to be stored.

In the following, some details regarding the proposed new set of reduced tables will be described. In accordance with the present invention, it is proposed to replace the previously-used tables "arith_cf_ng_hash[]" and "arith_cf_ng[][]" by the new set of tables given in Figs. 14 and 16(1) to 16(32). The new tables "arith_cf_ng_hash[]" and "arith_cf_ng[pki][545]" exhibit reduced sizes compared to the previously used tables (for example, tables used in the USAC reference model 0) and are subsequently called "reduced tables". The memory requirements of the noiseless coder (encoder or decoder) using the reduced tables are listed in detail in the table of Fig. 8.

The new set of tables shows a size reduction factor of about 7 compared to the original ones. The reduction was achieved by reducing the number of probability distribution

models and by optimizing the selected models. Further, the mapping between the context state and the newly-defined models was optimized.

In the following, a performance evaluation in terms of bitrates will be given. In particular, a lossless transcoding between a bitstream, which was generated using the previously applied "large" tables, and a transcoded bitstream, which is provided in accordance with the proposed "smaller" tables will be discussed. The new set of tables was proven to be able to be transcode transparently a bitstream generated with the previously-used tables, which are also designated as "reference model 0 tables" or "RM0 tables". The transcoding was performed using the transcoding scheme described in Fig. 9.

Fig. 9 shows a block schematic diagram of a lossless transcoding from the "reference model 0 tables" to the "reduced tables". As can be seen from Fig. 9, the evaluation setup comprises a USAC RM0 encoder 910, which receives, as an input information, spectral values 908, and which provides an arithmetically encoded representation 912 of the spectral values using the "old" tables of the reference model 0. Accordingly, a so-called "RM0" bitstream is obtained, which comprises the arithmetically-encoded spectral values 912. The evaluation setup 900 also comprises lossless transcoding 920, in which the arithmetically-encoded spectral values 912 of the RM0 bitstream are decoded by an entropy decoder using the "old" tables (reference model 0 tables) to obtain decoded spectral values 924. The lossless transcoding also comprises an entropy encoder, which is configured to encode the decoded spectral values 924 using the "new" reduced tables in order to obtain a reduced-tables bitstream 928. Subsequently, the RM0 bitstream, which comprises the representation of the spectral values encoded using the "old" tables, is compared with the reduced-tables bitstream 928, which comprises the representation of the spectral values encoded using the "new" tables.

For verification purposes, the so-called RM0 bitstream is decoded using a USAC reference decoder and using the "old" tables to obtain a so-called RM0 synthesis result 942. Also, the so-called reduced-table bitstream 928 is decoded using a USAC reference decoder and using the "new" tables. Accordingly, a reduced-tables synthesis result 952 is obtained. The RM0 synthesis result 942 is subsequently compared with the reduced-table synthesis result 952 to verify the correctness of the implementation.

In the following, some results of the analysis will be described. The tables of Figs. 10 and 11 display minimum, maximum and average bitrates over all sub-segments of the entire, concatenated, encoded item with the RM0 tables and the reduced table respectively. For each operating mode, an individual sub-segment length was determined by a combination

of succeeding access-units, whose length is closest to 100 ms. Sub-segment lengths and according bitrates are specified in the two tables. As mentioned earlier, the lossless transcoding from the RM0 tables bitstream to the reduced tables bitstream is achieved for each operating mode, i.e. the bit reservoir conditions were not violated while obtaining a bit-exact synthesis. The table of Fig.12 compares the bitrates generated only by the core coder when using the RM0 tables and the reduced tables. The reduced tables perform on average slightly better than the RM0 tables for each operating mode except at 64 kbit per second (kbps) stereo, where the average increase of bits is only 0.02% of 64 kbps, which corresponds to approximately 0.5 bits/frame. Nevertheless, the bitstream generated with the reduced tables still matches the bitrate requirement at this operating mode.

The table of Fig. 13 shows, for each of the operating modes, the worse and best cases of the difference of gathered bits after and before transcoding the RM0 tables bitstream to the reduced tables bitstream. The difference is looked on a sub-segment basis. It can be observed in all the cases that the performance of the reduced tables is very consistent and very stable. For a sub-segment, the maximum increase of bits by replacing the RM0 tables with the reduced tables is below 6 % of the total bitrate. On the other hand, the decrease of bits can reach more than 6 %.

To summarize the above, new tables for the USAC spectral noiseless coding have been proposed, the sizes of which are significantly reduced while maintaining the high coding performance of the spectral noiseless coding module. The achieved size reduction is of about a ratio of 7 or more than 90 kWords. The proposed new set of tables permits to significantly reduce the memory requirements and therefore lowers the implementation complexity. A bit exactness is maintained for each operating mode when comparing the synthesized output wave forms.

The above-mentioned advantages are reached by improving a specific part of the arithmetic coding. In some embodiments, a new hash table, arith_cf_ng_hash[128], a new set of 32 probability models, arith_cf_ng[32][545], and the corresponding mapping function "arith_get_pk()" are used. The update of the arithmetic coder tables (when compared to previously-used arithmetic coder tables) changes the mapping of the state index s into the probability model index pki as well as the probability models themselves. It does not change neither the computation of the state index s, nor the way the probability model is used afterwards for coding the current symbol (i.e. the group index ng of the present 4-tuple).

The main advantage of the new tables is to reduce the memory requirements for storing the tables, which have now a size of about 15 kWords (i.e. 15*1024 words of 32 bits each) instead of about 110 kWords, while maintaining the coding efficiency.

One of the important aspects, which brings along the above-discussed improvements, is the inventive mapping between state-index and probability-model index. This mapping is done by calling the function arith_get_pk() with the state variable t as an input argument.

The return value is the probability model index pki, which is used by the arithmetic coder as the probability distribution (also called cumulative-frequencies) for coding the current symbol (or to select the appropriate cumulative-frequencies table out of a set of cumulative-frequencies-tables).

The mapping is done in two steps, which are done in two different parts of the function.

In the first part 540d (of the function "arith_get_pk()"), the hash table "arith_cf_ng_hash" (also designated with ari_pk_hash[]) is considered and used for checking, if the current state t is a significant state. A significant state is a state, which was previously selected during a training phase to have its "own" mapping into pki. The non-significant states are mapped later on into pki in the second part 550e of the function arith_get_pk() by using default mappings (also designated as range-based mapping). According to the invention, the number of significant states was reduced to 67, which is drastically lower than the previous 22955 significant states from the old table. By reducing the number of significant states, the size of the table, "arith_cf_ng_hash[]" (also designated as ari_pk_hash[]), is proportionally reduced. It can also be expected that the performance will be affected. However, the performance was maintained by assuring that the training was done correctly. The training was performed for both AAC and TCX in a switching coding structure. Such precautions were not taken when generating the old tables and the retained 67 states are the most useful states for a non-switching-audio coder and also for a switching audio coder.

The following code permits to detect if the present state t is a significant state. If yes, the function is finished (because the return-statement is reached) in returns the associated probability model index pki:

```
i=63*t;
    for (;;)
```

```
{
        j=ari_pk_hash[i&127];
        if ( j==0xFFFFFFFFul ) break;
        if ( (j>>8)==t ) return j&255;
        i++;
}
```

The coefficient 63 in the first line of the above code (see also reference numeral 540c) determines the starting point in the search of the hash table. It is fixed during the training phase by using an exhaustive global optimization. Inside the hash table, the state index t is coded on the 24 "last" bits of the entries (for example the 24 leftmost or most-significant bits of the entries). The "first" bits (for example the eight rightmost or least-significant bits) correspond to the associated probability index. In a straightforward implementation, the table would contain only 67 entries. However, for minimizing the number of access to the table, an escape mechanism is used. Indeed, 128-67=61 escape symbols of value 0xFFFFFFFFul are inserted and permit to reduce the number of accesses to the table when a state is insignificant. In this case, when such a symbol is met, the code jumps directly into a second part 540e of the function "arith_get_pk()" (because the "break" statement is reached in this case).

The second part of 540e of the mapping function is used for mapping the non-significant states. The states are divided into seven non-uniform sections according to the indices. Each section is associated to a probability model. The mapping is recorded in the lookup table psci[]. The bits beyond the 22th bit of t indicate the accuracy of the level prediction. According to the prediction accuracy, a different mapping is used. In total, seven sections and four accuracies are considered, which correspond to a total of 28 different mappings stored in psci[]. The mapping is done as follows:

```
p=psci+7*(t>>22);
    j= t & 4194303;
    if ( j<436961 )
    {
        if ( j<252001 ) return p[(j<243001)?0:1]; else return
p[(j<288993)?2:3];
    }
    else
    {
        if ( j<1609865 ) return p[(j<880865)?4:5]; else return p[6];
```

}

Finally, the probability models were reduced from 128 to 32. It was found that a lot of the models previously used were only seldom selected and not really useful. An appropriate training permits to select only 32 representative models.

To summarize the above, the embodiments according to the invention are related to the above-discussed mapping tables, which are implemented in an audio encoder and an audio decoder. The advantage of the new tables partly comes from the adequate training, which was performed. Further, the invention is based on the considerations about the best dimension of the tables.

7. Bitstream Syntax

In the following, the bitstream syntax of a bitstream carrying the arithmetically-encoded spectral information will be described taking reference to Figs. 6a to 6h.

Fig. 6a shows a syntax representation of s so-called USAC raw data block ("usac_raw_data_block()").

The USAC raw data block comprises one or more single channel elements ("single_channel_element()") and/or one or more channel pair elements ("channel_pair_element()").

Taking reference now to Fig. 6b, the syntax of a single channel element is described. The single channel element comprises a linear-prediction-domain channel stream ("lpd_channel_stream ()") or a frequency-domain channel stream ("fd_channel_stream ()") in dependence on the core mode.

Fig. 6c shows a syntax representation of a channel pair element. A channel pair element comprises core mode information ("core_mode0", "core_mode1"). In addition, the channel pair element may comprise a configuration information "ics_info()". Additionally, depending on the core mode information, the channel pair element comprises a linear-prediction-domain channel stream or a frequency-domain channel stream associated with a first of the channels, and the channel pair element also comprises a linear-prediction-domain channel stream or a frequency-domain channel stream associated with a second of the channels.

The configuration information "ics_info()", a syntax representation of which is shown in Fig. 6d, comprises a plurality of different configuration information items, which are not of particular relevance for the present invention.

A frequency-domain channel stream ("fd_channel_stream ()"), a syntax representation of which is shown in Fig, 6e, comprises a gain information ("global_gain") and a configuration information ("ics_info ()"). In addition, the frequency-domain channel stream comprises scale factor data ("scale_factor_data ()"), which describes scale factors used for the scaling of spectral values of different scale factor bands, and which is applied, for example, by the scaler 150 and the rescaler 240. The frequency-domain channel stream also comprises arithmetically-coded spectral data ("ac_spectral_data ()"), which represents arithmetically-encoded spectral values.

The arithmetically-coded spectral data ("ac_spectral_data()"), a syntax representation of which is shown in Fig. 6f, comprises an optional arithmetic reset flag ("arith_reset_flag"), which is used for selectively resetting the context, as described above. In addition, the arithmetically-coded spectral data comprise a plurality of arithmetic-data blocks ("arith_data"), which carry the arithmetically-coded spectral values. The structure of the arithmetically-coded data blocks depends on the number of frequency bands (represented by the variable "num_bands") and also on the state of the arithmetic reset flag, as will be discussed in the following.

The structure of the arithmetically-encoded data block will be described taking reference to Fig. 6g, which shows a syntax representation of said arithmetically-coded data blocks. The data representation within the arithmetically-coded data block depends on the number lg of spectral values to be encoded, the status of the arithmetic reset flag and also on the context, i.e. the previously-encoded spectral values.

The context for the encoding of the current set of spectral values is determined in accordance with the context determination algorithm shown at reference numeral 660. The arithmetically-encoded data block comprises lg/4 sets of codewords, each set of codewords representing a tuple of spectral values. A set of codewords comprises an arithmetic codeword "acod_ng [pki][ng]" representing a group index ng of the tuple of spectral values using between 1 and 20 bits. A set of codewords also comprises an arithmetic codeword "acod_ne[ne]" representing an element index ne of a tuple of spectral values if the group comprising the tuple of spectral values includes more than one element. In addition, the set of codewords comprises one or more codewords "acod_r[][][][]" if the tuple of spectral values requires more bit planes than the most-significant bit plane for a correct

representation. The codeword "acod_ne [ne]" represents the element index using between 1 and 20 bits and the codeword "acod_r [][][][]" represents a less-significant bit plane using between 1 and 20 bits.

If, however, one or more less-significant bit-planes are required (in addition to the most-significant bit plane) for a proper representation of the tuple of spectral values, this is signaled by using one or more arithmetic escape codewords ("ARITH_ESCAPE"). Thus, it can be generally said that for a tuple of spectral values, it is determined how many bit planes (the most-significant bit plane and, possibly, one or more additional less-significant bit planes) are required. If one or more less-significant bit planes are required, this is signaled by one or more arithmetic escape codewords "acod_ng [pki][ARITH_ESCAPE]", which are encoded in accordance with a currently-selected cumulative-frequencies-table, a cumulative-frequencies-table-index of which is given by the variable pki. In addition, the context is adapted, as can be seen at reference numeral 664, 662, if one or more arithmetic escape codewords are included in the bitstream. Following the one or more arithmetic escape codewords, an arithmetic codeword "acod_ng [pki][ng]" is included in the bitstream, as shown at reference numeral 663, wherein pki designates the currently-valid probability model index (taking into consideration the context adaptation caused by the inclusion of the arithmetic escape codewords), and wherein ng designates the group index associated with the most-significant bit plane of the tuple of spectral values to be encoded. The group index can be derived, in an encoder, by evaluating the table dg vectors, which allows for deriving the group index ng and an element index ne associated to a tuple of spectral values when taken in combination with the table "dgroups".

If the group, which includes the tuple of spectral values to be encoded, comprises more than one element, the arithmetically-encoded data block comprises the codeword "acod_ne[ng]" encoding the element index ne using an appropriately-selected cumulative-frequencies-table.

As discussed above, the presence of any less-significant-bit planes results in the presence of one or more codewords "acod_r [][][][]", each of which represents a tuple of 4 bits of a least-significant bit plane. The one or more codewords "acod_r[][][][]" are encoded in accordance with a corresponding cumulative-frequencies-table, which is constant and context-independent.

In addition, it should be noted that the context is updated after the encoding of each tuple of spectral values, as shown at reference numeral 668, such that the context is typically different for encoding of two subsequent tuples of spectral values.

Fig. 6h shows a legend of definitions and help elements defining the syntax of the arithmetically-encoded data block.

To summarize the above, a bitstream format has been described, which may be provided by the audio coder 100, and which may be evaluated by the audio decoder 200. The bitstream of the arithmetically-encoded spectral values is encoded such that it fits the decoding algorithm discussed above.

In addition, it should be generally noted that the encoding is the inverse operation of the decoding, such that it can generally be assumed that the encoder performs a table lookup using the above-discussed tables, which is approximately inverse to the table lookup performed by the decoder. Generally, it can be said that a man skilled in the art who knows the decoding algorithm and/or the desired bitstream syntax will easily be able to design an arithmetic encoder, which provides the data defined in the bitstream syntax and required by the arithmetic decoder.

8. Decoding Method

In the following, a method for providing a decoded audio information on the basis of an encoded audio information will be described taking reference to Fig. 20. The method 2000 comprises a first step 2010 of providing a plurality of decoded spectral values on the basis of an arithmetically-encoded representation of the spectral values. The method 2000 further comprises a second step 2020 of providing a time-domain audio representation using the decoded spectral values. The step 2010 comprises selecting 2012 a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on a state index. The step 2010 also comprises a sub-step 2014 of applying the selected cumulative-frequencies-table to derive a group-index from a variable-length-codeword representing the group index. The step 2010 also comprises the sub-step 2016 of deriving values of a most-significant bit-plane of a tuple of spectral values using the group index and the element index, the element index designating an element within a group selected by the group index. The step 2010 also comprises a sub-step 2018 of providing a tuple of decoded spectral values using the values of the most-significant bit plane of the tuple of spectral values.

9. Encoding Method

In the following, a method for providing an encoded audio information on the basis of an input audio information will be discussed taking reference to Fig. 21. The method 2100 of Fig. 21 comprises a first step of providing a frequency-domain audio representation on the basis of a time-domain audio representation of the input audio information, such that the frequency-domain audio representation comprises a set of spectral values, and such that an energy is compacted in a sub-set of the spectral values. The method 2100 also comprises a second step 2120 of encoding a tuple of adjacent spectral values of the set of spectral values, or of encoding a tuple of adjacent spectral values of a pre-processed version of the set of spectral values. The step 2120 comprises a sub-step 2122 of mapping values of a most-significant bit plane of a tuple of spectral values onto a group index and an element index, the element index designating an element within a group selected by the group index. The step 2120 also comprises a sub-step 2124 of selecting a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on a state index describing a state of the arithmetic encoder. The step 2120 also comprises a sub-step 2126 of arithmetically encoding the group index using the selected cumulative-frequencies-table (selected in the sub-step 2124) in order to obtain an arithmetically-encoded variable-length codeword.

It should be noted here that the methods 2000 and 2100 of Figs. 20 and 21 can be supplemented by any of the features and functionalities described herein with respect to the inventive coding scheme. In addition, the methods 2000 and 2100 can be supplemented by any of the features and functionalities of the apparatus described herein. Also, the coding tables described herein are preferable used in connection with the encoding method and decoding method.

10. Implementation Alternatives

Although some aspects have been described in the context of an apparatus, it is clear that these aspects also represent a description of the corresponding method, where a block or device corresponds to a method step or a feature of a method step. Analogously, aspects described in the context of a method step also represent a description of a corresponding block or item or feature of a corresponding apparatus. Some or all of the method steps may be executed by (or using) a hardware apparatus, like for example, a microprocessor, a programmable computer or an electronic circuit. In some embodiments, some one or more of the most important method steps may be executed by such an apparatus.

The inventive encoded audio signal can be stored on a digital storage medium or can be transmitted on a transmission medium such as a wireless transmission medium or a wired transmission medium such as the Internet.

Depending on certain implementation requirements, embodiments of the invention can be implemented in hardware or in software. The implementation can be performed using a digital storage medium, for example a floppy disk, a DVD, a Blue-Ray, a CD, a ROM, a PROM, an EPROM, an EEPROM or a FLASH memory, having electronically readable control signals stored thereon, which cooperate (or are capable of cooperating) with a programmable computer system such that the respective method is performed. Therefore, the digital storage medium may be computer readable.

Some embodiments according to the invention comprise a data carrier having electronically readable control signals, which are capable of cooperating with a programmable computer system, such that one of the methods described herein is performed.

Generally, embodiments of the present invention can be implemented as a computer program product with a program code, the program code being operative for performing one of the methods when the computer program product runs on a computer. The program code may for example be stored on a machine readable carrier.

Other embodiments comprise the computer program for performing one of the methods described herein, stored on a machine readable carrier.

In other words, an embodiment of the inventive method is, therefore, a computer program having a program code for performing one of the methods described herein, when the computer program runs on a computer.

A further embodiment of the inventive methods is, therefore, a data carrier (or a digital storage medium, or a computer-readable medium) comprising, recorded thereon, the computer program for performing one of the methods described herein.

A further embodiment of the inventive method is, therefore, a data stream or a sequence of signals representing the computer program for performing one of the methods described herein. The data stream or the sequence of signals may for example be configured to be transferred via a data communication connection, for example via the Internet.

A further embodiment comprises a processing means, for example a computer, or a programmable logic device, configured to or adapted to perform one of the methods described herein.

A further embodiment comprises a computer having installed thereon the computer program for performing one of the methods described herein.

In some embodiments, a programmable logic device (for example a field programmable gate array) may be used to perform some or all of the functionalities of the methods described herein. In some embodiments, a field programmable gate array may cooperate with a microprocessor in order to perform one of the methods described herein. Generally, the methods are preferably performed by any hardware apparatus.

The above described embodiments are merely illustrative for the principles of the present invention. It is understood that modifications and variations of the arrangements and the details described herein will be apparent to others skilled in the art. It is the intent, therefore, to be limited only by the scope of the impending patent claims and not by the specific details presented by way of description and explanation of the embodiments herein.

While the foregoing has been particularly shown and described with reference to particular embodiments above, it will be understood by those skilled in the art that various other changes in the forms and details may be made without departing from the sprit and cope thereof. It is to be understood that various changes may be made in adapting to different embodiments without departing from the broader concept disclosed herein and comprehended by the claims that follow.

Claims

1. An audio decoder (200) for providing a decoded audio information (212) on the basis of an encoded audio information (210), the audio decoder comprising:

an arithmetic decoder (230) for providing a plurality of decoded spectral values (232; a,b,c,d) on the basis of an arithmetically-encoded representation (222; acod_ng[pki][ng], acod_ne[ne], acod_r[][][][]) of the spectral values; and

a frequency-domain to time-domain converter (260) for providing a time-domain audio representation (262, 212) using the decoded spectral values (232; a,b,c,d), in order to obtain the decoded audio information (262, 212);

wherein the arithmetic decoder (230) is configured to derive a group index (ng) from a variable-length-code word (acod_ng[pki][ng]) representing the group index (ng) in dependence on a state index (s);

wherein the arithmetic decoder (230) is configured to derive values (a, b, c, d) of a most-significant bit-plane of a tuple of spectral values using the group index (ng) and an element index (ne), the element index (ne) describing an element within a group selected by the group index;

wherein the arithmetic decoder (230) is configured to provide a tuple of decoded spectral values using the values of the most-significant bit-plane of the tuple of spectral values; and

wherein the arithmetic decoder is configured to select a cumulative-frequencies-table (arith_cf_ng[pki][545]) out of a set of 32 cumulative-frequencies-tables (arith_cf_ng[0][545] to arith_cf_ng[31][545]) in dependence on the state-index (t), and to apply the selected cumulative-frequencies-table to derive the group-index (ng) from the variable-length codeword (acod_ng[pki][ng]) representing the group-index (ng).

2. The audio decoder (200) according to claim 1,

wherein the arithmetic decoder is configured to derive a 7-bit hash-table index value (i & 127) from the state-index (t), and to obtain a hash-table value (j) from a

hash-table (ari_pk_hash[]; arith_cf_ng_hash[]), which hash-table comprises a mapping of 128 hash-table-index-values onto corresponding hash-table-entry-values, and

wherein the arithmetic decoder is configured to decide whether the hash-table value (j) associated to the hash-table index-value derived from state-index (t) is an escape value (0xFFFFFFFF), a valid cumulative-frequencies-table-identifier-value associated to the state-index or an invalid cumulative-frequencies-table-identifier-value in conflict with the state-index, and to derive the cumulative frequencies-table index-value (pki) from the hash-table-value, if the hash-table-value associated to the hash-table index-value derived from state-index (t) is a valid cumulative frequencies-table-identifier-value associated to the state index (t), and to provide a cumulative-frequencies-table index-value (pki) in dependence on an identification of an interval of values in which the state index (t) is contained, if the hash-table-value (j)) associated to the hash-table index-value derived from state-index (t) is the escape value; and

wherein the arithmetic decoder is configured to scan through the entries of the hash-table (ari_pk_hash[]; arith_cf_ng_hash[]), starting from the entry designated by the hash-table index-value (i & 127) derived from the state-index (t), until the hash-table value (j) associated to the hash-table index-value derived from state-index (t) is either an escape value (0xFFFFFFFF) or a valid cumulative-frequencies-table-identifier-value associated to the state-index, and

wherein the arithmetic decoder is configured to provide a cumulative-frequencies-table index-value (pki) in dependence on an identification of an interval of values in which the state index (t) is contained, if the hash-table-value reached when scanning through the entries of the hash-table is the escape value (0xFFFFFFFF), and to derive the cumulative-frequencies-table index-value (pki) from the hash-table-value reached when scanning through the entries of the hash-table, if the hash-table-value reached when scanning through the entries of the hash-table is a valid cumulative-frequencies-table identifier-value associated to the state index (t).

3. **The audio decoder (200) according to claim 2, wherein the hash-table** (ari_pk_hash[]; arith_cf_ng_hash[]) is configured to map 67 values of the 7-bit hash-table-index value (i & 107) onto valid cumulative-frequencies-table identifier-

values, and to map 61 values of the 7-bit hash-table-index value (i & 107) onto the escape value (0xFFFFFFFF).

4. The audio decoder (200) according to claim 3, wherein the arithmetic decoder (230) is configured to map 67 different values of the state index (t) onto 26 different cumulative-frequencies-table index-values (pki), such that 26 different cumulative-frequencies-table index-values are associated to 67 different significant states described by the state index (t).

5. The audio decoder (200) according to one of claims 2-4, wherein the arithmetic decoder (230) is configured to map a plurality of non-significant states onto 9 different cumulative-frequencies-table index-values.

6. The audio decoder(200) according to one of claims 1-5,

wherein the arithmetic decoder (230) is configured to set

i = 63 *t

and to iteratively perform the algorithm:

```
j=ari_pk_hash[i&127];
if ( j==0xFFFFFFFFul ) break;
if ( (j>>8)==t ) return j&255;
i++;
```

until the first condition j == 0xFFFFFFFF

or the second condition (j >> 8) == t

is fulfilled,

wherein s designates the state index,

wherein i and j designate integer variables,

wherein ari_pk_hash[i & 128] designates an entry of index (i & 128) of a hash-table,

wherein "&" designates a bit-wise logical AND operator,

wherein ">> 8" designates a binary shift-to-the-right operation shifting by 8 bits,

wherein "==" designates a check for an identity condition,

wherein "++" designates an increase-by-one operator,

wherein "return j & 255" designates an operation of returning a value described by the 8 least significant bits of variable j as a cumulative-frequencies-table-index value (pki); and

wherein the arithmetic decoder is configured to perform the algorithm

```
p=psci+7*(t>>22);
j= t & 4194303;
if ( j<436961 )
{
        if ( j<252001 )
         return p[(j<243001)?0:1];
        else
          return p[(j<288993)?2:3];
}
else
{
        if ( j<1609865 )
         return p[(j<880865)?4:5];
        else
         return p[6];
}
```

in response to a break condition, wherein

```
psci[28] = {
        24,5,25,26,27,28,29,30,5,30,30,30,30,31,5,5,
        5,5,5,5,5,5,5,5,5,5,5,5
};
```

defines an array of different cumulative frequencies-table-index-values having array indices of 0 to 27, and

wherein the operation p = psci + 7 ·t >> 22 sets the pointer p to an element having an element index which is determined by 7 times a value represented by the most significant bits of the state index t,

wherein an operation of the forms "return p[condition?x:y]" returns an entry of the array psci having an element index which is determined by a sum of 7 times a value represented by the most significant bits of the state index s and the value x, if the condition is fulfilled and returns an entry of the array psci having an element index which is determined by a sum of 7 times a value represented by the most significant bits of the state index s and the value y, if the condition is not fulfilled, and

when the hash-table ari_pk_hash is defined given in Fig. 14.

7. The audio decoder (200) according to one of claims 1 to 6,

wherein the arithmetic decoder (230) is configured to obtain (550a) a codeword-value (value) in dependence on a plurality of bits (acod_ng[pki][ng]) of the encoded audio information (222),

to obtain (550b) a relative position value (cum) describing a relative position of the codeword value (value) within a range between a lower range boundary value (low) and a higher range boundary value (high),

to determine (550c) in which interval (cum_freq[symbol], cum_freq[symbol-1]) out of a plurality of intervals defined by entries of the selected cumulative-frequencies-table (cum_freq, arith_cf_ng[pki][545]) the relative position value (cum) is included,

to provide (550d) a symbol information (symbol) in response to a result of determining, in which interval out of the plurality of intervals defined by the entries of the selected cumulative-frequencies-table (cum_freq, arith_cf_ng[pki][545]) the relative position value (cum) is included, and

to update (550e) one or both of the range boundary values (high, low) in dependence on one or more of the entries (cum_freq[symbol-1], cum_freq[symbol]) of the selected cumulative-frequencies-table (cum_freq, arith_cf_ng[pki][545]) associated with the symbol information (symbol), and

to rescale (550fa) the range between the range boundary values (low, high), and

to update (550fb) the codeword value (value) using one or more additional bits (arith_get-next_bit) of the encoded audio information.

8. The audio decoder (200) according to one of claims 1-7, wherein the arithmetic decoder is configured to obtain the group-index (ng) on the basis of the encoded audio information (222) using a selected one (arith_cf_ng[pki][545]) of the 32 cumulative frequencies-tables (arith_cf_ng[0][545] to arith_cf_ng[31][545]), and using the following algorithm arith_decode(),

```
arith_decode()
{
        if(arith_first_symbol())
        {
                value = 0;
                for (i=1; i<=20; i++)
                {
                        value = (value<<1) | arith_get_next_bit();
                }
                low=0;
                high=1048575;
        }

        range = high-low+1;
        cum =((((int64) (value-low+1))<<16)-((int64) 1))/((int64) range);
        p = cum_freq-1;

        do
        {
                q=p+(cfl>>1);
                if ( *q > cum ) { p=q; cfl++; }
                cfl>>=1;
```

```
        }
        while ( cfl>1 );

        symbol = p-cum_freq+1;
        if(symbol)
            high=low+(((int64) range)*((int64)cum_freq[symbol-1]))>>16 - 1;

        low  += (((int64) range)* ((int64) cum_freq[symbol]))>>16;

        for (;;)
        {
            if ( high<524286)
            {  }

            else if ( low>=524286)
            {
                value -=524286;
                low -=524286;
                high -=524286;
            }

            else if ( low>=262143 && high<786429)
            {
                value -= 262143;
                low -= 262143;
                high -= 262143;
            }

            else break;

            low += low;
            high += high+1;
            value = (value<<1) | arith_get_next_bit();
        }

        return symbol;
    }
```

wherein the arithmetic decoder is configured to initialize a codeword value "value" in dependence on 20 bits (acod_ng]pki][ng]) of the encoded audio information, if a helper function "arith_first_symbol()" indicates that a first symbol of a sequence of symbols is decoded;

wherein a helper function "arith_get_next_bit()" provides a next bit of the encoded audio information,

wherein an operator "<<" designates a Boolean shift-to-the-left-operation to shift to the left bits of an operand preceding the operator "<<" by a number of bits specified by an operand following the operator "<<",

when the operator "|" designates a Boolean or-operation,

wherein the operator "(int64)" designates that a number type to be used for a representation of a following operand is a 64-bit integer number type,

wherein "cum_freq" is an address value of a first entry of a selected cumulative-frequencies-table (arith_cf_ng[pki][545]),

wherein "*q" designates a (q-cum_freq+1)-th entry of the selected cumulative frequencies-table (arith_cf_ng[pki][545]) having an entry index of q-cum_freq ,

wherein cfl is a variable initialized to a length of the selected cumulative-frequencies-table (arith_cf_ng[pki][545]) and modified during a processing of the algorithm "arith_decode()",

wherein "cum_freq[symbol-1]" designates a symbol-th entry of the selected cumulative frequencies-table (arith_cf_ng[pki][545]); and

wherein an operation "for(;;){...}" indicates to repeat a block of instructions included in the brackets "{" and "}" until a "break" instruction is reached.

9. The audio decoder according to claim 8, wherein the 32 cumulative frequencies-tables (arith_cf_ng[pki=0][545] to arith_cf_ng[pki=31][545]) are associated to 32 cumulative-frequencies-table index-values pki between 0 and 31, and

wherein the cumulative frequencies-tables are defined in accordance with the table representations of Fig. 16(1) to Fig. 16(32).

10. The audio decoder (200) according to one of the claims 1-9, wherein the arithmetic decoder (230) is configured to determine a number (mm) of group elements of a group designated by the group-index (ng),

to evaluate an encoded representation (acod_ne[ne]) of an element index, to obtain a decoded representation of the element index, if the group designated by the group index (ne) comprises more than one element,

to determine a look-up address base (4·(og + ne)) in dependence on the group-index (ng) and, if the group designated by the group-index (ng) comprises more than one element, the element index (ne), and

to determine look-up values of the most-significant bit plane in a look-up table (dgvectors) in dependence on the look-up address base.

11. The audio decoder (200) according to one of claims 1 to10, wherein the arithmetic decoder (230) is configured to determine the state index (t) for decoding a current tuple of spectral values using an algorithm "arith_get_context" defined as follows:

```
arith_get_context()
{
        t0=q[0][i].v+1;
        t1=q[1][i-1].v+1;
        t2=q[0][i-1].v+1;
        t3=q[0][i+1].v+1;

        if ( (t0<10) && (t1<10) && (t2<10) && (t3<10) ){
                if ( t2>1 ) t2=2;
                if ( t3>1 ) t3=2;
                return 3*(3*(3*(3*(3*(10*(10*t0+t1))+t2)+t3)));
        }
        if ( (t0<34) && (t1<34) && (t2<34) && (t3<34) ){
```

```
        if ( (t2>1) && (t2<10) )
            t2=2;
        else if ( t2>=10 )
            t2=3;

        if ( (t3>1) && (t3<10) )
            t3=2;
        else if ( t3>=10 )
            t3=3;

        return 252000+4*(4*(34*(34*t0+t1))+t2)+t3;
    }

    if ( (t0<90) && (t1<90) ) return 880864+90*(90*t0+t1);

    if ( (t0<544) && (t1<544) ) return 1609864 + 544*t0+t1;

    if ( t0>1 )
    {
        a0=q[0][i].a;
        b0=q[0][i].b;
        c0=q[0][i].c;
        d0=q[0][i].d;
    }
    else a0=b0=c0=d0=0;

    if ( t1>1 )
    {
        a1=q[1][i-1].a;
        b1=q[1][i-1].b;
        c1=q[1][i-1].c;
        d1=q[1][i-1].d;
    }
    else a1=b1=c1=d1=0;

    l=0;
    do
```

```
        {
            a0>>=1;
            b0>>=1;
            c0>>=1;
            d0>>=1;

            a1>>=1;
            b1>>=1;
            c1>>=1;
            d1>>=1;

            l++;
        }
        while (     (a0<-4) || (a0>=4) || (b0<-4) || (b0>=4) ||
                    (c0<-4) || (c0>=4) || (d0<-4) || (d0>=4) ||
                    (a1<-4) || (a1>=4) || (b1<-4) || (b1>=4) ||
                    (c1<-4) || (c1>=4) || (d1<-4) || (d1>=4)      );


        if ( t0>1 ) t0=1+(egroups[4+a0][4+b0][4+c0][4+d0] >> 16);
        if ( t1>1 ) t1=1+(egroups[4+a1][4+b1][4+c1][4+d1] >> 16);

        return 1609864 +  ((l<<24)|(544*t0+t1));
    }
```

wherein q[0][i].v designates a context variable of a tuple of spectral values of a previous audio frame associated with the same frequencies like the current tuple of spectral values;

wherein q[1][i-1] designates a context variable of a tuple of spectral values of a current audio frame associated with lower frequencies than the current tuple of spectral values;

wherein q[0][i-1] designates a context variable of a tuple of spectral values of the previous audio frame associated with lower frequencies than the current tuple of spectral values;

wherein q[0][i+1] designates a context variable of a tuple of spectral values of the previous audio frame associated with higher frequencies than the current tuple of spectral values;

wherein "&&" designates a logical AND operation;

wherein q[0][i].a designates a first tuple value "a" of a tuple of spectral values of the previous audio frame associated with the same frequencies as the current tuple of spectral values;

wherein q[0][i].b designates a second tuple value "b" of a tuple of spectral values of the previous audio frame associated with the same frequencies as the current tuple of spectral values;

wherein q[0][i].c designates a third tuple value "c" of a tuple of spectral values of the previous audio frame associated with the same frequencies as the current tuple of spectral values;

wherein q[0][i].d designates a fourth tuple value "d" of a tuple of spectral values of the previous audio frame associated with the same frequencies as the current tuple of spectral values;

wherein q[1][**i-1**].a designates a first tuple value "a" of a tuple of spectral values of the current audio frame associated with lower frequencies than the current tuple of spectral values;

wherein q[1][**i-1**].b designates a second tuple value "b" of a tuple of spectral values of the current audio frame associated with lower frequencies than the current tuple of spectral values;

wherein q[1][**i-1**].c designates a third tuple value "c" of a tuple of spectral values of the current audio frame associated with lower frequencies than the current tuple of spectral values;

wherein q[1][**i-1**].d designates a fourth tuple value "d" of a tuple of spectral values of the previous audio frame associated with lower frequencies than the current tuple of spectral values;

wherein a0, b0, c0, d0, a1, b1, c1, d1 are variables representing a signed number in a 2-s complement representation;

wherein ">>" designates a logical shift-to-the right operator; and

wherein "egroups[4+a0][4+b0][4+c0][4+d0]" defines an entry having entry indices 4+a0, 4+b0, 4+c0, 4+d0 of a four-dimensional array "egroups",

wherein "egroups[4+a1][4+b1][4+c1][4+d1]" defines an entry having entry indices 4+a1, 4+b1, 4+c1, 4+d1 of the four-dimensional array "egroups",

wherein the array "egroups" is defined in accordance with the table representations of Fig. 19(1) to Fig. 19(32).

12. The audio decoder (200) according to claim 11,

wherein the arithmetic decoder (230) is configured to update the context variables using the following algorithm:

```
arith_update_context()
{
        q[1][i].a=a;
        q[1][i].b=b;
        q[1][i].c=c;
        q[1][i].d=d;

        if ( (a<-4) || (a>=4) || (b<-4) || (b>=4) ||
             (c<-4) || (c>=4) || (d<-4) || (d>=4) )
        {
              q[1][i].v =1024;
        }
        else q[1][i].v=egroups[4+a][4+b][4+c][4+d];

        if(i==lg/4 && core_mode==1){
              qs[0]=q[1][0];
```

```
                ratio= ((float) lg)/((float)1024);
                for(j=0; j<256; j++)
                {
                        k = (int) ((float) j*ratio);
                        qs[1+k] = q[1][1+j] ;
                }
                qs[previous_lg/4+1] = q[1][lg/4+1];
                previous_lg = 1024;
        }

        if(i==lg/4 && core_mode==0)
        {
                for(j=0; j<258; j++)
                {
                        qs[j] = q[1][j];
                }
                previous_lg = min(1024,lg);
        }
}
```

wherein, a, b, c, d designate values of a current, completely-decoded tuple of spectral values;

wherein lg/4 designates a number of 4 tuples associated to the current audio frame;

wherein "core_mode = = 1" indicates a linear-prediction-domain core mode; and

wherein "core_mode = = 0" indicates a frequency-domain core mode;

wherein a "(float)" operator indicates to use a floating point number representation; and

wherein a "(int)" operator indicates to use an integer number representation.

13. An audio encoder (100) for providing an encoded audio information (112) on the basis of an input audio information (110), the audio encoder comprising:

an energy-compacting time-domain to frequency-domain converter (130) for providing a frequency-domain audio representation (132) on the basis of a time-domain representation (110, 110a) of the input audio information, such that the frequency-domain audio representation (1332) comprises a set of spectral values;

an arithmetic encoder (170) configured to encode a tuple (a,b,c,d) of adjacent spectral values (132), or a pre-processed version (152) thereof, using a variable-length codeword (acod_ng[pki][ng], acod_ne[ne], acod_r[][][][]);

wherein the arithmetic encoder (170) is configured to map values of a most-significant bit-plane (a>>lev, b>>lev, c>>lev, d>>lev) of a tuple (a,b,c,d) of spectral values onto a group-index (ng) and an element index (ne), the element index (ne) describing an element within a group selected by the group index (ng);

wherein the arithmetic encoder (170) is configured to select a cumulative-frequencies-table (arith_cf_ng[pki][545]) out of a set of 32 cumulative-frequencies-tables (arith_cf_ng[0][545] to arith_cf_ng[31][545]) in dependence on a state index (t) of the arithmetic encoder; and

wherein the arithmetic encoder (170) is configured to arithmetically encode the group index (ng) using the selected cumulative-frequencies-table (arith_cf_ng[pki][545]) in order to obtain an arithmetically-encoded variable-length-codeword (acod_ng[pki][ng]).

14. A method (2000) for providing a decoded audio representation on the basis of an encoded audio representation, the method comprising:

providing (2010) a plurality of decoded spectral values on the basis of an arithmetically-encoded representation of the spectral values; and

providing (2020) a time-domain audio representation using the decoded spectral values;

wherein providing (2010) a plurality of decoded spectral values on the basis of an arithmetically-encoded representation of the spectral values comprises

selecting (2012) a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on a state index,

applying (2014) the selected cumulative-frequencies-table to derive a group-index from a variable-length-codeword representing the group index,

deriving (2016) values of a most-significant bit-plane of a tuple of spectral values using the group index and an element index, the element index designating an element within a group selected by the group index; and

providing (2018) a tuple of decoded spectral values using the values of the most-significant bit-plane of the tuple of spectral values.

15. A method (2100) for providing an encoded audio representation on the basis of an input audio representation, the method comprising:

providing (2110) a frequency-domain audio representation on the basis of a time-domain audio representation of an input audio information, such that the frequency-domain audio representation comprises a set of spectral values, and such that an energy is compacted in a sub-set of the spectral values; and

encoding (2120) a tuple of adjacent spectral values of the set of spectral values, or of a pre-processed version of the set of spectral values, wherein the encoding of the tuple of adjacent spectral values comprises

mapping (2122) values of a most-significant bit-plane of the tuple of spectral values onto a group index and an element index, the element index designating an element within a group selected by the group index,

selecting (2124) a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on a state index describing a state of the arithmetic encoding, and

arithmetically encoding (2126) the group index using the selected cumulative-frequencies-table, in order to obtain an arithmetically-encoded variable length-codeword.

16. A computer program for performing the method according to claim 14 or 15, when the computer program runs on a computer.

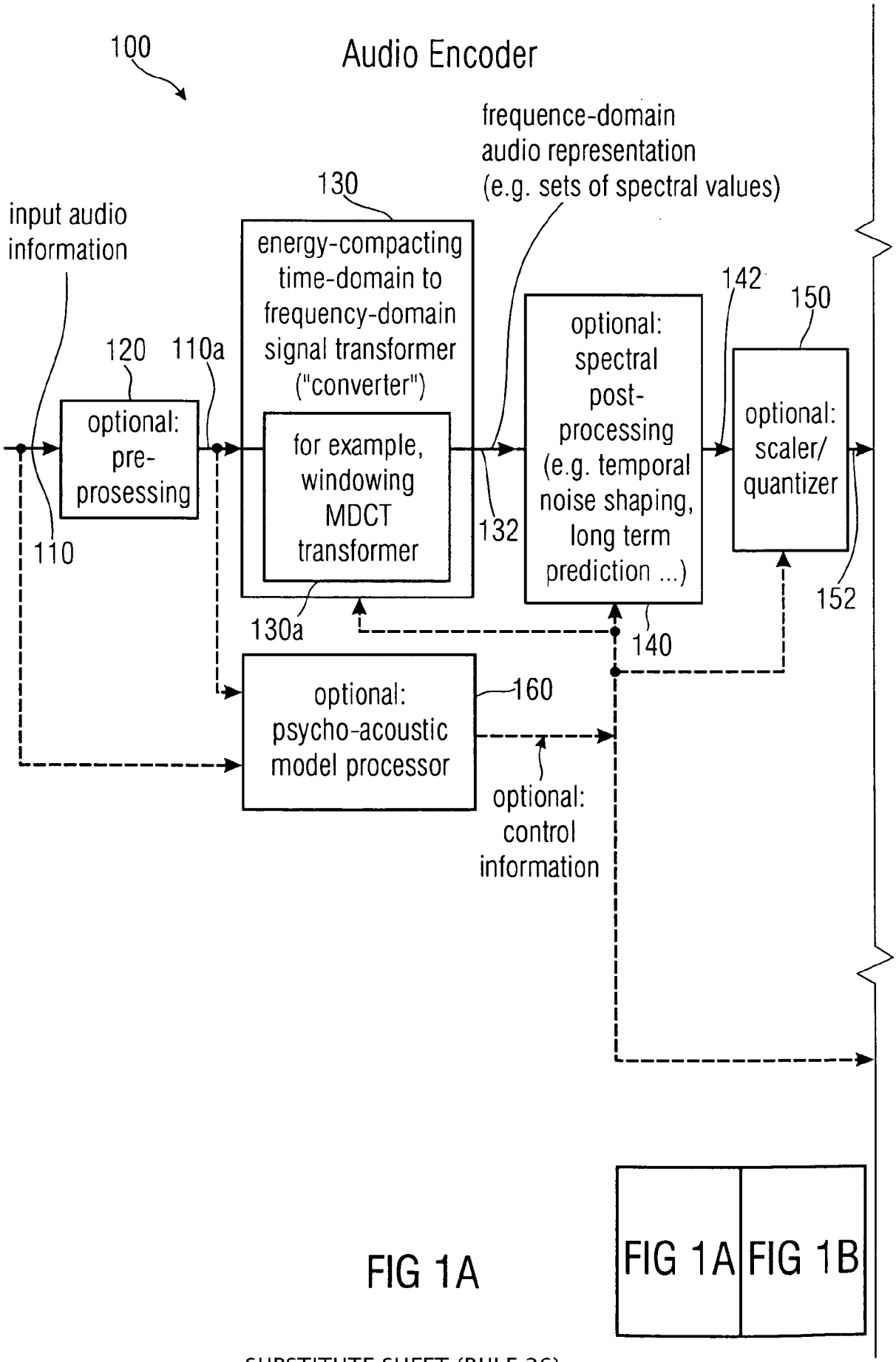


FIG 1A

FIG 1A	FIG 1B

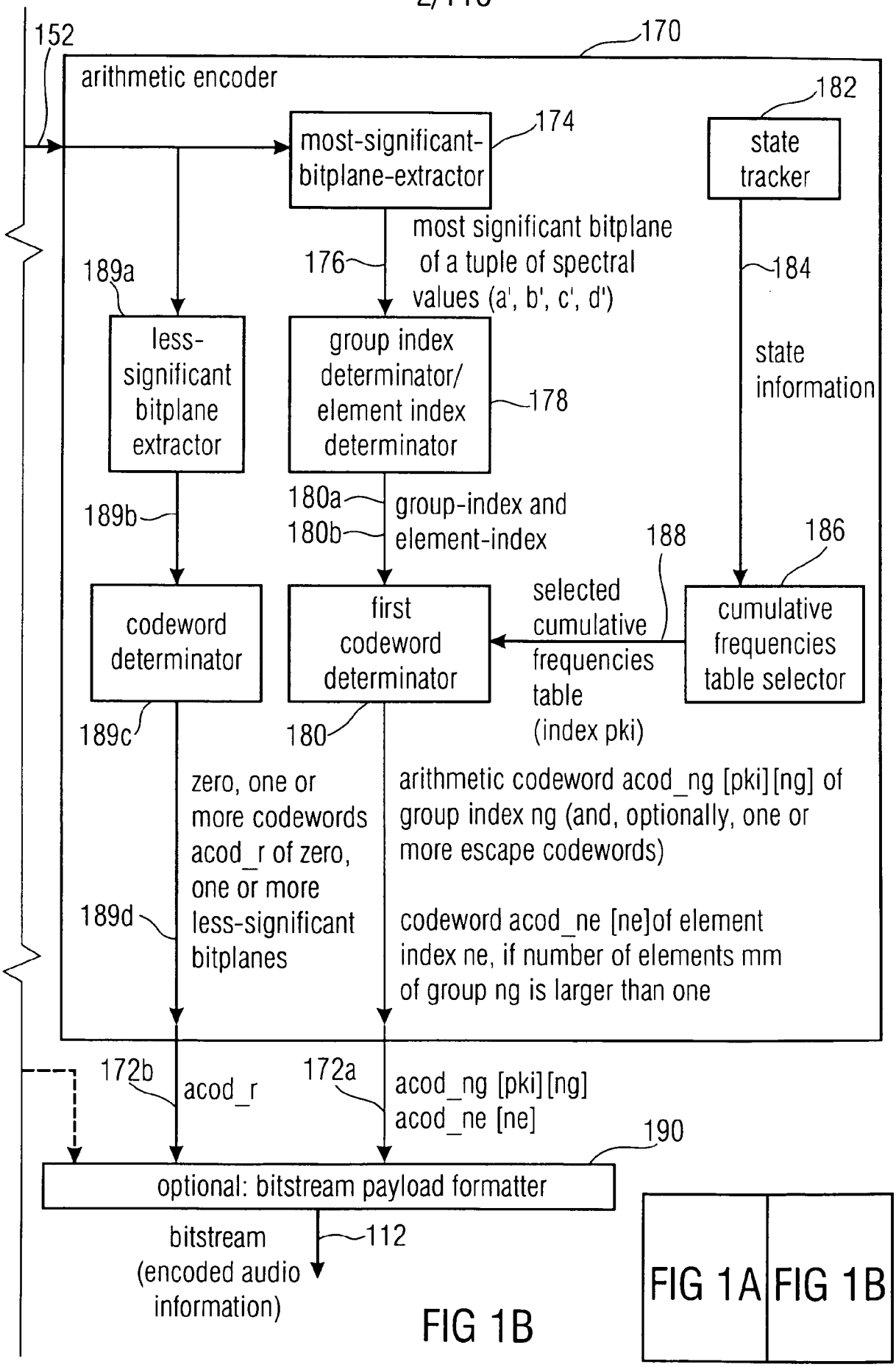

152
170
arithmetic encoder
182
174
most-significant-bitplane-extractor
state tracker
most significant bitplane of a tuple of spectral values (a', b', c', d')
176
184
189a
less-significant bitplane extractor
group index determinator/ element index determinator
178
state information
180a
180b
group-index and element-index
189b
188
186
codeword determinator
first codeword determinator
selected cumulative frequencies table (index pki)
cumulative frequencies table selector
189c
180
zero, one or more codewords acod_r of zero, one or more less-significant bitplanes
arithmetic codeword acod_ng [pki][ng] of group index ng (and, optionally, one or more escape codewords)
189d
codeword acod_ne [ne]of element index ne, if number of elements mm of group ng is larger than one
172b
acod_r
172a
acod_ng [pki][ng]
acod_ne [ne]
190
optional: bitstream payload formatter
bitstream (encoded audio information)
112
FIG 1A
FIG 1B

FIG 1B

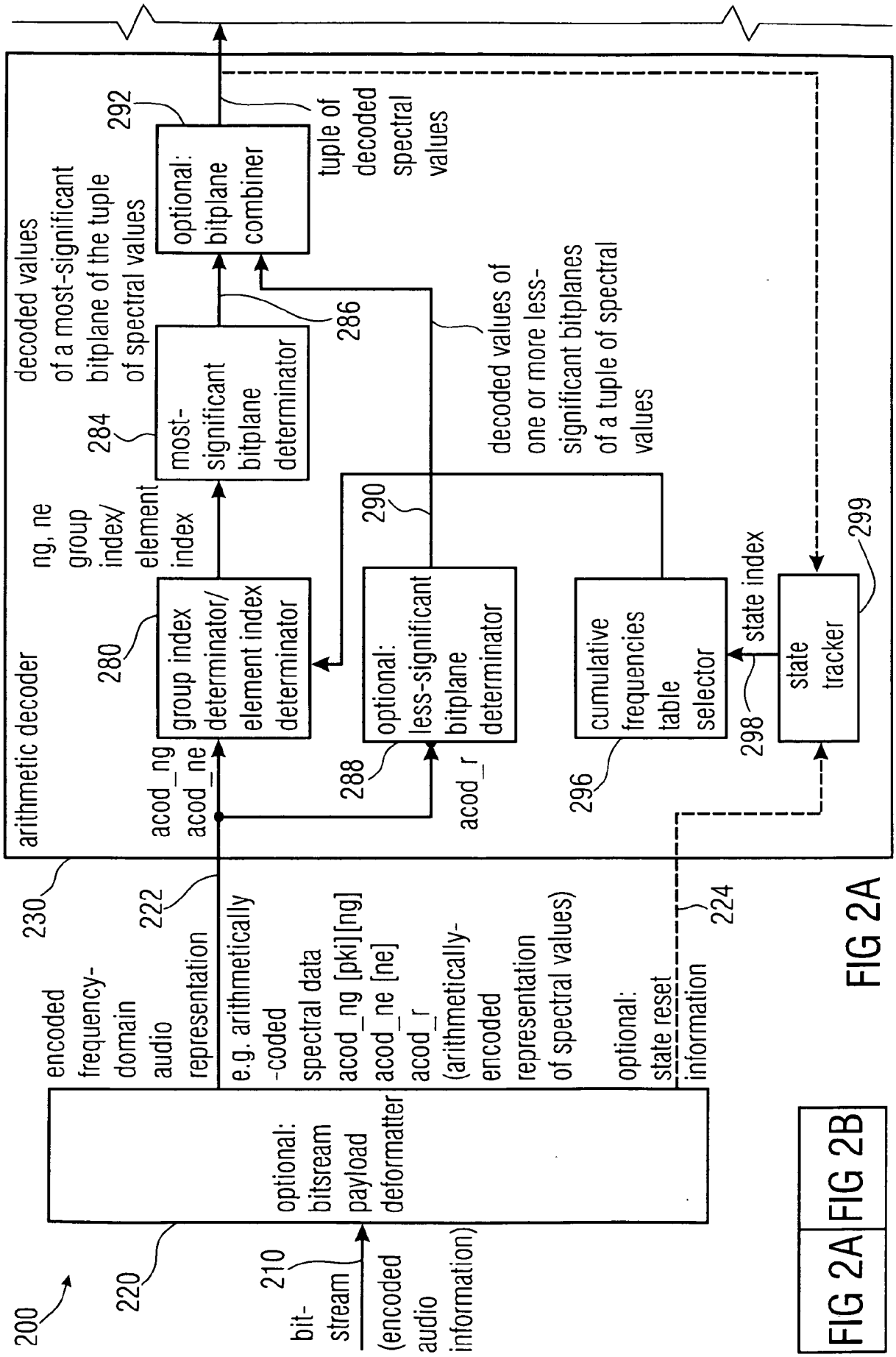


FIG 2A

FIG 2A FIG 2B

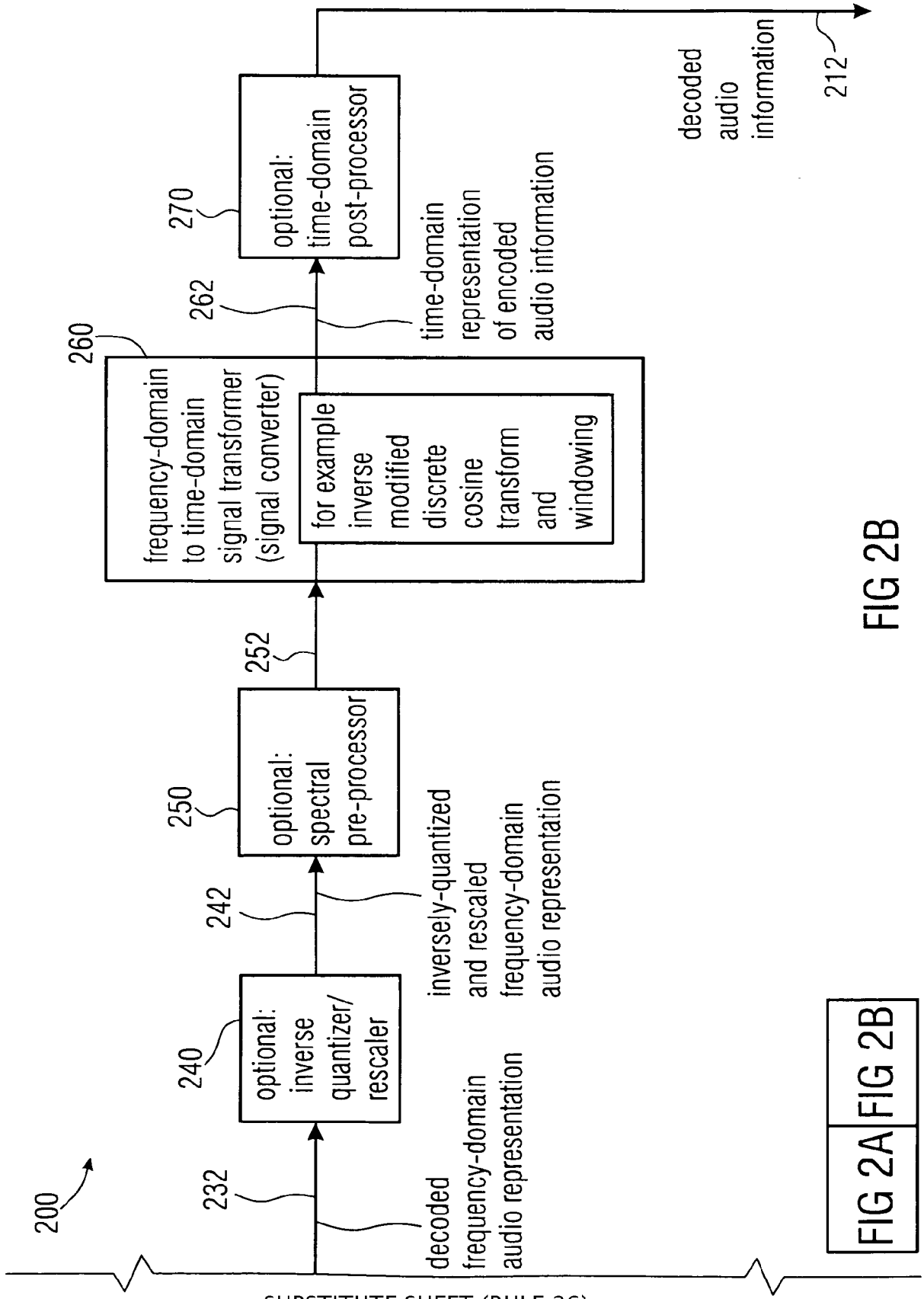

200
232
decoded frequency-domain audio representation
240
optional: inverse quantizer/ rescaler
242
inversely-quantized and rescaled frequency-domain audio representation
250
optional: spectral pre-processor
252
260
frequency-domain to time-domain signal transformer (signal converter)
for example inverse modified discrete cosine transform and windowing
262
time-domain representation of encoded audio information
270
optional: time-domain post-processor
212
decoded audio information
FIG 2A
FIG 2B

FIG 2B

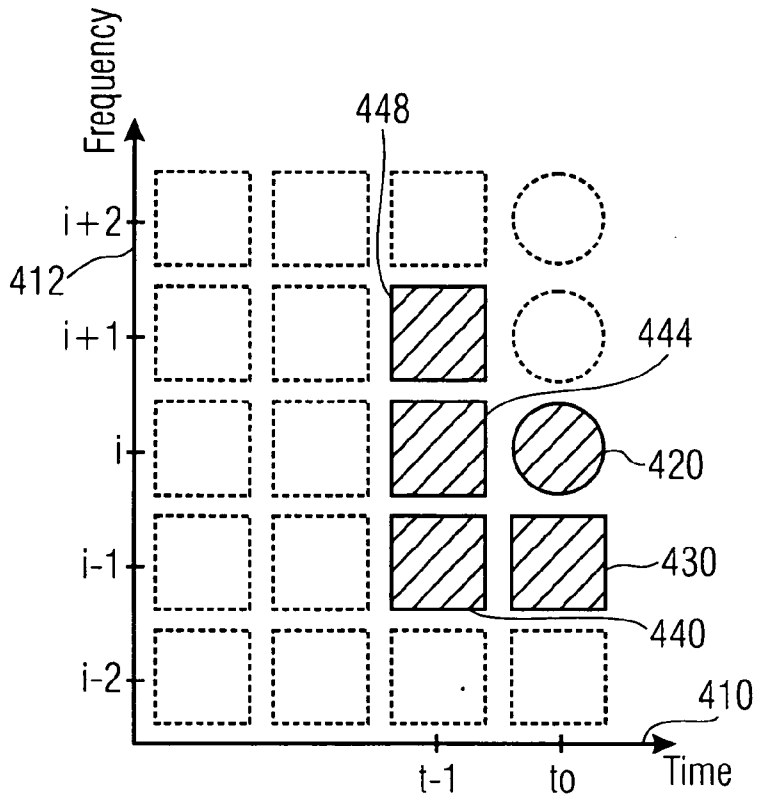
```
tuples_decode ()
{
  310  if (arith_reset_flag) {
           arith_reset_context();
       }
       esle {
           arith_map_context(lg/4);
       }
  312  for (i=1); i≤lg/4; i++) {
  312a     s = arith_get_context (i);
           lev = s>>24;
           t = s & 0xFFFFFF + 1
  312b     for (j=0;;) {
  312ba        pki = arith_get_pk(t);
               cum_freq = table_start_position (pki);
               cfl = table_lenght (pki) ;
               ng = arith_decode();                          uses bits acod_ng
               if ( ng != ARITH_ESCAPE)                      of bitstream
                   break;
               lev += 2;
               if (!j) { t+= 4194304; } else { t=0; }
               j=1;
           }

  312c     j = dgroups[ng];
           mm = j & 255;
           ne = 0
           if (mm>1){
  312ca        cum_freq = arith_cf_ne+((mm * (mm-1))>>1)[];
               cfl = mm;
               ne = arith_decode();                          uses bits acod_ne
           }                                                 of bitstream
  312d     a=dgvectors[4*(j>>8+ne)];
           b=dgvectors[4*(j>>8+ne)+1];
           c=dgvectors[4*(j>>8+ne)+2];
           d=dgvectors[4*(j>>8+ne)+3];

  312e     for (l=lev; l>0; l--){

               cum_freq=arith_cf_r[];
               cfl=16;
               r=arith_decode;                               uses bits acod_r
               a = (a<<1) | (r&1)                            of bitstream
               b = (b<<1) | ((r>>1)&1)
               c = (c<<1) | ((r>>2)&1)
               d = (d<<1) | (r>>3)
           }
  314      arith_update_context(a,b,c,d,i,lg/4);
       }
}
```

FIG 3



Context for the state calculation

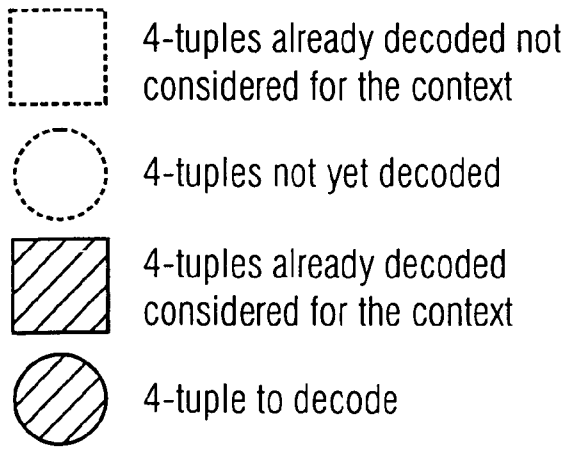


FIG 4

```
/*global variables*/
q[2][290] /*current context*/
qs[258] /*past context*/
previous_lg/4 /*number of 4-tuples of the past context*/

arith_reset_context()
{

        for(i=0;i<258;i++){
                qs[i].a=0; qs[i].b=0; qs[i].c=0; qs[i].d=0
                qs[i].v=-1;
        }
        for(i=0;i<290;i++){
                q[0][i].a=0; q[0][i].b=0; q[0][i].c=0; q[0][i].d=0
                q[0][i].v=-1;
                q[1][i].a=0; q[1][i].b=0; q[1][i].c=0; q[1][i].d=0
                q[1][i].v=-1;

        }
        previous_lg=1024;
}
```

FIG 5A

```
/*input variable*/
lg/4 /*number of 4-tuples*/

arith_map_context()
{
        v=w=0

        if(core_mode==1){
                q[0][v++]=qs[w++];
        }

        ratio= ((float)previous_lg)/((float)lg);
        for(j=0; j<lg/4; j++){
                k = (int) ((float)) ((j)*ratio);
                q[0][v++] = qs[w+k];
        }

        if(core_mode==0){
                q[0][lg/4]=qs[previous_lg/4];
        }
        q[0][lg/4+1]=qs[previous_lg/4+1];

        previous_lg=lg;
}
```

FIG 5B

```
/*input variables*/
i /*the index of the 4-tuple to decode in the vector*/

arith_get_context()
{
530a    t0=q[0] [    i].v+1;
        t1=q[1] [    i-1].v+1;
        t2=q[0] [    i-1].v+1;
        t3=q[0] [    i+1].v+1;

        if ( (t0<10) && (t1<10) && (t2<10) && (t3<10) ){
530b            if ( t2>1 ) t2=2;
                if ( t3>1 ) t3=2;
                return 3*(3*(3*(3*(3*(10*(10*t0+t1))+t2)+t3)));
        }
        if ( (t0<34) && (t1<34) && (t2<34) && (t3<34) ){

530c            if ( (t2>1) && (t2<10) ) t2=2; else if ( t2>=10 ) t2=3;
                if ( (t3>1) && (t3<10) ) t3=2; else if ( t3>=10 ) t3=3;
                return 252000+4*(4*(34*(34*t0+t1))+t2)+t3;
        }
530d    if ( (t0<90) && (t1<90) ) return 880864+90*(90*t0+t1);
530e    if ( (t0<544) && (t1<544) ) return 1609864 + 544*t0+t1;

        if ( t0>1 )
        {
                a0=q[0][i].a;
                b0=q[0][i].b;
                c0=q[0][i].c;
                d0=q[0][i].d;
530fa   }
        else a0=b0=c0=d0=0;

        if ( t1>1 )
        {
                a1=q[1][i-1].a;
                b1=q[1][i-1].b;
                c1=q[1][i-1].c;
                d1=q[1][i-1].d;
        }
        else a1=b1=c1=d1=0;

530f    l=0;
        do
        {
                a0>>=1;
                b0>>=1;
                c0>>=1;
                d0>>=1;

530fb           a1>>=1;
                b1>>=1;
                c1>>=1;
                d1>>=1;

                l++;
        }
        while (    (a0<-4) || (a0>=4) || (b0<-4) || (b0>=4) || (c0<-4) ||
(c0>=4) || (d0<-4) || (d0>=4) ||
                (a1<-4) || (a1>=4) || (b1<-4) || (b1>=4) || (c1<-4) || (c1>=4)
|| (d1<-4) || (d1>=4)    );

530fc   if ( t0>1 ) t0=1+(egroups[4+a0][4+b0][4+c0][4+d0] >> 16);
        if ( t1>1 ) t1=1+(egroups[4+a1][4+b1][4+c1][4+d1] >> 16);

530fd   return 1609864 +  ((l<<24)|(544*t0+t1));
}
```

FIG 5C

```
/*input variable*/
t /* State of the context*/

arith_get_pk (t)
{
        psci[28] = {
540a
        24,5,25,26,27,28,29,30,5,30,30,30,30,31,5,5,
        5,5,5,5,5,5,5,5,5,5,5,5
};
540b    register unsigned char *p;
        register unsigned long    i,j;
540c    i=63*t;
        for (;;)
        {
540da   j=ari_pk_hash[i&127];
540db   if ( j==0xFFFFFFFFul ) break;
540d    if ( (j>>8)==t ) return j&255;
540dc   i++;
540dd   }
540ea   p=psci+7*(t>>22);
        j= t & 4194303;
540eb   if ( j<436961 )
        {
540e            if ( j<252001 ) return p[(j<243001)?0:1]; else return
        p[(j<288993)?2:3];
        }
540ec   else
        {
                if ( j<1609865 ) return p[(j<880865)?4:5]; else return p[6];
        }
}
```

FIG 5D

```
/*helper funtions*/
bool arith_first_symbol(void);
        /* Return TRUE if it is the first symbol of the sequence, FALSE
otherwise*/
Ushort arith_get_next_bit(void);
        /* Get the next bit of the bitstream*/

/* global variables */
low
high
value

/* input variables */
cum_freq[];  /* cumulative frequencies table*/
cfl;                /* length of cum_freq[] */

arith_decode()
{
        if(arith_first_symbol())
        {
                value = 0;
                for (i=1; i<=20; i++)
                {
550a            value = (val<<1) | arith_get_next_bit();
                }
                low=0;
                high=1048575;
        }

        range = high-low+1;
550b    cum =((((int64) (value-low+1))<<16)-((int64) 1))/((int64) range);
        p = cum_freq-1;

        do
        {
                q=p+(cfl>>1);
550c            if ( *q > cum ) {p=q; cfl++; }
                cfl>>=1;
        }
        while ( cfl>1 );

550d    symbol = p-cum_freq+1;
        if(symbol)
                high=low+(((int64) range)*((int64)cum_freq[symbol-1]))>>16 - 1;
550e
        low  += (((int64) range)* ((int64) cum_freq[symbol]))>>16;

        for (;;)
        {
        if ( high<524286) { }
            else if ( low>=524286)
                {
                        value -=524286;
                        low -=524286;
                        high -=524286;
550fa           }
550f            else if ( low>=262143 && high<786429)
                {
                        value -= 262143;
                        low -= 262143;
                        high -= 262143;
                }
        else break;

        low += low;
550fb   high += high+1;
                value = (value<<1) | arith_get_next_bit();
        }
        return symbol;
}
```

FIG 5E

```
mm = dgroups[ng]&255
og = dgroups[ng]>>8
```

FIG 5F

```
a = dgvectors[4*(og+ne)]
b = dgvectors[4*(og+ne)+1]
c = dgvectors[4*(og+ne)+2]
d = dgvectors[4*(og+ne)+3]
```

FIG 5G

```
a = (a<<1) | (r&1)
b = (b<<1) | ((r>>1)&1)
c = (c<<1) | ((r>>2)&1)
d = (d<<1) | (r>>3)
```

FIG 5H

```
/*input variables*/
a,b,c,d /* value of the decoded 4-tuple */
i /*the index of the 4-tuple to decode in the vector*/
lg/4 /*number of 4-tuples*/

arith_update_context()
{
        q[1][   i].a=a;
580a    q[1][   i].b=b;
        q[1][   i].c=c;
        q[1][   i].d=d;
        if ( (a<-4) || (a>=4) || (b<-4) || (b>=4) || (c<-4) || (c>=4) || (d<-
4) || (d>=4) ) {
580b            q[1][   i].v =1024;
        }
        else q[1][   i].v=egroups[4+a][4+b][4+c][4+d];

        if(i==lg/4 && core_mode==1){
                qs[0]=q[1][0];

                ratio= ((float) lg)/((float)1024);
580c            for(j=0; j<256; j++){
                        k = (int) ((float) j*ratio);
                        qs[1+k] = q[1][1+j] ;
                }
                qs[previous_lg/4+1] = q[1][lg/4+1];
                previous_lg = 1024;
        }
        if(i==lg/4 && core_mode==0){
                for(j=0; j<258; j++){
580d                    qs[j] = q[1][j];
                }
                previous_lg = min(1024,lg);
        }
}
```

FIG 5I

4.2. Definitions

(a,b,c,d)	4-tuple to decode
ng	Group index of the most significant 2-bits-wise plane of the 4-tuple, where 0 <= ng <= 544. The last value 544 corresponds to the escape symbol, ARITH_ESCAPE.
ne	Element index within a group. ne lies between 0 and the cardinal of each group mm. The maximum number of element within a group is 73.
lev	Level of the remaining bit-planes. It corresponds to number the bit planes less significant than the most significant 2 bits-wise plane.
egroups[a][b][c][d]	Group index table. It permits to map the most significant 2 bits-wise plane of the 4-tuple (a,b,c,d) into the 544 groups.
mm	Cardinal of the group
og	Offset of the group
dgroups[]	Maps the group index ng to the cardinal of each group mm (first 8 bits) and the offset of the group og in dgvectors[] (last 8bits).
dgvectors[]	Map the offset of the group og and the index of the element ne to the most significant 2 bits-wise plane of the 4-tuple (a,b,c,d).
arith_cf_ng_hash[]	Hash table mapping the state of context to a cumulative frequencies table index pki.
arith_cf_ng[pki][545]	Models of the cumulative frequencies for the group index symbol ng.
arith_cf_ne[]	Cumulative frequencies for element index symbol ne.
r	Bit plane of the 4-tuple less significant than the most significant 2-bits wise plane.
arith_cf_r []	Cumulative frequencies for the least significant bit-planes symbol r
previous_lg	number of transmitted spectral coefficients previously decoded by the arithmetic decoder

FIG 5J

```
usac_raw_data_block()
{
  single_channel_element (); and/or
  channel_pair_element ();
}
```

FIG 6A

Syntax of single_channel_element()

Syntax	No. of bits	Mnemonic
single_channel_element()		
{		
core_mode	1	uimsbf
if (core_mode==1){		
lpd_channel_stream();		
}		
else{		
fd_channel_stream();		
}		
}		

FIG 6B

Syntax of channel_pair_element()

Syntax	No. of bits	Mnemonic
channel_pair_element()		
{		
core_mode0	1	unimsbf
core_mode1	1	unimsbf
ics_info();	optional: common ics_info for two channels	
if (core_mode0==1){		
lpd_channel_stream();		
}		
else{		
fd_channel_stream();		
}		
if (core_mode1==1){		
lpd_channel_stream();		
}		
else{		
fd_channel_stream();		
}		
}		

FIG 6C

Syntax of ics_info()

Syntax	No. of bits	Mnemonic
ics_info()		
{		
window_length;	1	uimsbf
if (window_length !=0){		
transform_length;	1	uimsbf
}		
else{		
transform_length=0;		
}		
window_shape;	1	uimsbf
if (window_length != 0 && transform_length != 0) {		
max_sfb;	4	uimsbf
scale_factor_grouping;	7	uimsbf
}		
else{		
max_sfb;	6	uimsbf
}		
}		

optional

FIG 6D

Syntax of fd_channel_stream()

Syntax		No. of bits	Mnemonic
fd_channel_stream()			
{			
global_gain;		8	**uimsbf**
ics_info();	(unless included in channel pair element)		
scale_factor_data();			
ac_spectral_data();			
}			

FIG 6E

Syntax of ac_spectral_data()

Syntax	No. of bits	Mnemonic
ac_spectral_data()		
{		
arith_reset_flag	1	uimsbf
for (win = 0; win < num_windows; win++){		
arith_data(num_bands, arith_reset_flag)		
}		
}		

FIG 6F

Syntax of arith__data()

Syntax	No. of bits	Mnemonic
arith_data(lg, arith_reset_flag)		
{		
660 if (arith_reset_flag) {		
arith_reset_context();		
}		
else {		
arith_map_context(lg/4);		
}		
for (i=0; i<lg/4; i++) {		
s = arith_get_context (i);		
lev = s>>24;		
t = s & 0xFFFFFF + 1;		
for (j=0;;) {		
662 pki = arith_get_pk(t)		
663 **acod_ng**[pki][ng]	1..20	vlclbf
if (ng != ARITH_ESCAPE)		
break;		
lev += 2;		
664 if (!j) { t += 4194304; } else { t=0; }		
j = 1;		
}		
j = dgroups[ng];		
mm = j & 255;		
if (mm>1) {		
acod_ne[ne]	1..20	vlclbf
}		
a = dgvectors[4*(j>>8+ne)]<<lev;		
b = dgvectors[4*(j>>8+ne)+1]<<lev;		
c = dgvectors[4*(j>>8+ne)+2]<<lev;		
d = dgvectors[4*(j>>8+ne)+3]<<lev;		
for (l=lev; l>0; l--) {		
acod_r[(a>>l)&1][(b>>l)&1][(c>>l)&1][(d>>l)&1]	1..20	vlclbf
}		
668 arith_update_context(a,b,c,d,i,lg/4);		
}		
}		

FIG 6G

Definitions

arith_data()	Data element to decode the spectral noiseless coder data
arith_reset_flag	Flag which indicates if the spectral noiseless context must be reset.
acod _ng[pki][ng]	Arithmetic codeword necessary for arithmetic decoding of the 4-tuple (a,b,c,d) group index ng, where a, b, c, d are the quantized spectral coefficients.
arith_ne[ne]	Arithmetic codeword necessary for arithmetic decoding of the 4-tuple (a,b,c,d) element index ne.
arith_r[][][][]	Arithmetic codeword necessary for arithmetic decoding of the 4-tuple (a,b,c,d) residual bit-planes.

Help elements

a,b,c,d	4-tuple corresponding to quantized spectral coefficients
lg	Number of quantized coefficients to decode.
i	Index of 4-tuples.
ng	Index of the group to which belongs the 4-tuple a,b,c,d
ne	Index of the 4-tuples a,b,c,d within the group ng
pki	Index of the cumulative frequencies table used by the arithmetic decoder for decoding ng.
arith_get_pk ()	Function that returns the index pki of cumulative frequencies table necessary to decode the codeword **acod _ng[pki][ng]**.
t	State of context
arith_get_context ()	Function that returns the state of the context.
arith_reset_context ()	Function that resets the context of the noiseless coder.
s	State of the context combined with predicted bit-plane level.
lev	Level of bit-planes to decode beyond the most significant 2-bits wise plane.
j	Indicate if an ARITH_ESCAPE symbol occurred
ARITH_ESCAPE	Escape symbol that indicates additional bit-planes to decode

FIG 6H

Table: Memory requirements of the USAC noiseless coder in term of words (32 bits)

Table Name	Description	Unit of Data	Memory (words of 32 bits)
arith_cf_ng_hash[32768]	Hash table mapping context to a probability distribution model, *pki*, of the group symbol, *ng*.	word	32 768
arith_cf_ng[256][545]	Cumulative frequencies of the group symbol for each probability distribution model *pki*.	1/2 word	69 760
egroups[8][8][8][8]	Group index of 4 tuple.	1/2 word	2 048
dgvectors[4*4096]	Map group index and element index to 4-tuple.	1/4 word	4 096
dgroups[544]	Map group index to cardinal of the group and offset in dgvectors	word	544
arith_cf_ne[2701]	Cumulative frequencies of the element index symbol	1/2 word	135.5
arith_cf_r[16]	Cumulative frequencies of least significant bit planes	1/2 word	8
Total			109 329.5

FIG 7

Table: Memory requirements of the USAC noiseless coder using the reduced tables

Table Name	Description	Unit of Data	Memory (words of 32 bits)
arith_cf_ng_hash[128]	Hash table mapping context to a probability model index.	word	128
arith_cf_ng[32][545]	Cumulative frequencies of groups for each probability distribution model	1/2 word	8720
egroups[8][8][8][8]	Group index of 4 tuple.	1/2 word	2048
dgvectors[4*4096]	Map group index and element index to 4-tuple.	1/4 word	4096
dgroups[544]	Map group index to cardinal of the group and offset in dgvectors	word	544
arith_cf_ne[2701]	Cumulative frequencies of the element index symbol	1/2 word	135.5
arith_cf_r[16]	Cumulative frequencies of least significant bit planes	1/2 word	8
Total			15 679.5

FIG 8

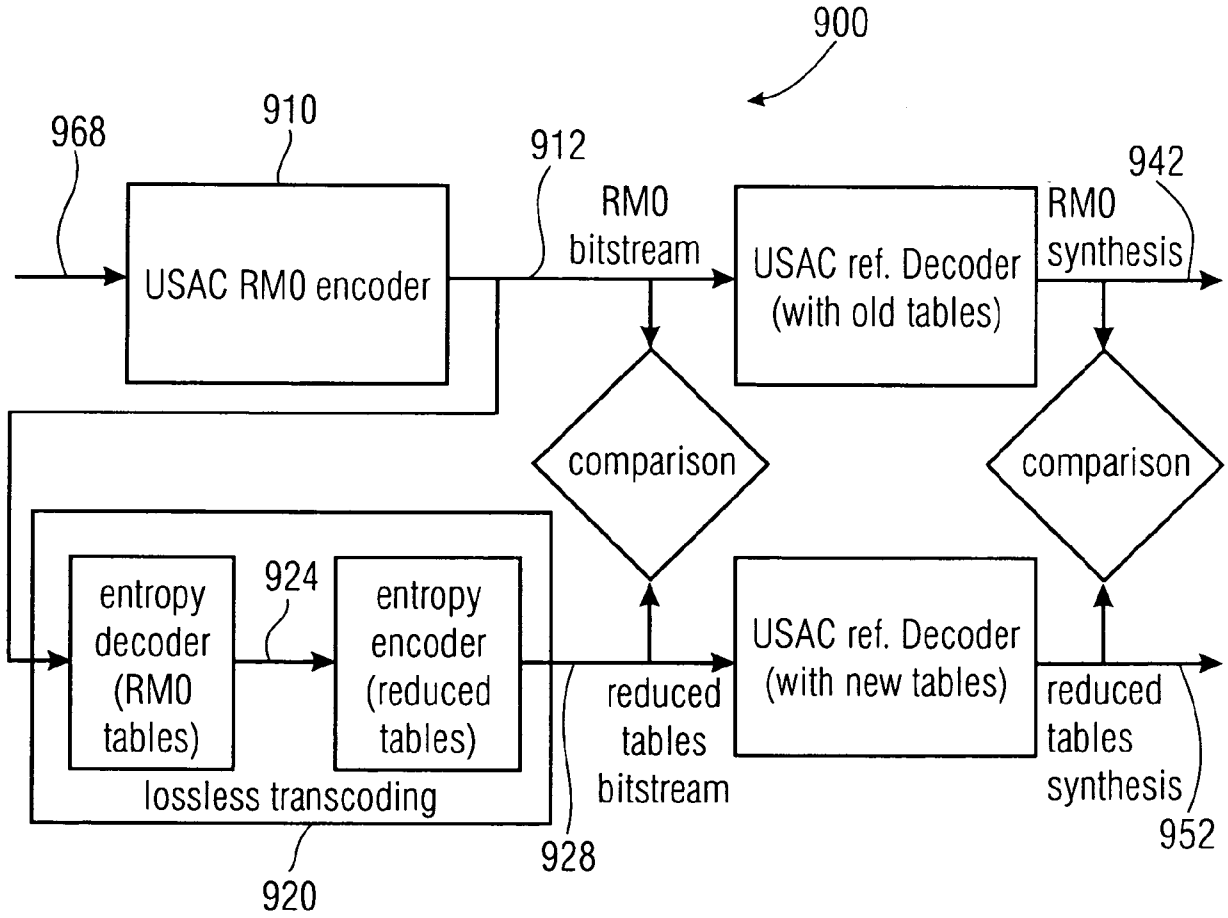

900
968
910
912
942
USAC RM0 encoder
RM0
bitstream
USAC ref. Decoder
(with old tables)
RM0
synthesis
comparison
comparison
entropy
decoder
(RM0
tables)
924
entropy
encoder
(reduced
tables)
lossless transcoding
920
reduced
tables
bitstream
928
USAC ref. Decoder
(with new tables)
reduced
tables
synthesis
952

FIG 9

Table: Minimum, maximum and average bitrates of USAC using RM0 tables

Operating Mode	Sub-Segment Length (ms)	Minimum Bitrate (kbps)	Maximum Bitrate (kbps)	Average Bitrate (kbps)
Test 1, 64kbps stereo	85.33	30.84	101.25	64.00
Test 2, 32kbps stereo	102.40	15.94	46.09	32.00
Test 3, 24kbps stereo	119.94	11.94	37.35	24.00
Test 4, 20kbps stereo	128.00	6.06	27.69	20.00
Test 5, 16kbps stereo	71.11	5.63	27.11	16.00
Test 6, 24kbps mono	119.94	6.14	35.88	24.00
Test 7, 20kbps mono	128.00	3.88	35.19	20.00
Test 8, 16kbps mono	71.11	11.59	25.76	16.00
Test 9, 12kbps mono	80.00	9.00	19.90	12.00

FIG 10

Table: Minimum, maximum and average bitrates of USAC using reduced tables

Operating Mode	Sub-Segment Length (ms)	Minimum Bitrate (kbps)	Maximum Bitrate (kbps)	Average Bitrate (kbps)
Test 1, 64kbps stereo	85.33	30.94	101.44	64.00
Test 2, 32kbps stereo	102.40	20.62	45.94	32.00
Test 3, 24kbps stereo	119.94	10.14	37.15	24.00
Test 4, 20kbps stereo	128.00	6.19	28.00	20.00
Test 5, 16kbps stereo	71.11	10.69	27.34	16.00
Test 6, 24kbps mono	119.94	6.07	35.55	24.00
Test 7, 20kbps mono	128.00	3.94	35.12	20.00
Test 8, 16kbps mono	71.11	11.81	25.99	16.00
Test 9, 12kbps mono	80.00	9.20	19.50	12.00

FIG 11

Table: Average bitrates produced by the core coder using RM0 and reduced tables

Operating mode	RM0 Tables (bits/s)	Reduced Tables (bits/s)	Difference after transcoding (bits/s)	Difference after transcoding (% of total bit-rate)
Test 1, 64kbps stereo	57545.00	57557.64	+12.64	+0.02
Test 2, 32kbps stereo	28109.12	28081.94	-27.18	-0.08
Test 3, 24kbps stereo	20589.39	20563.25	-26.14	-0.11
Test 4, 20kbps stereo	17258.12	17233.89	-24.23	-0.12
Test 5, 16kbps stereo	14047.03	14019.15	-27.88	-0.08
Test 6, 24kbps mono	21420.50	21394.53	-25.97	-0.11
Test 7, 20kbps mono	17689.25	17668.42	-20.83	-0.10
Test 8, 16kbps mono	14292.32	14270.02	-22.30	-0.14
Test 9, 12kbps mono	10839.74	10821.38	-18.36	-0.15

FIG 12

Table: Best and worst cases on a sub-segment basis

Operating modes	Sub-Segment Length (ms)	Best case		Worst Case	
		(bits/s)	(%)	(bits/s)	(%)
Test 1, 64kbps stereo	85.33	-1652.34	-2.58	+1863.28	+2.91
Test 2, 32kbps stereo	102.40	-996.09	-3.11	+742.19	+2.32
Test 3, 24kbps stereo	119.94	-725.35	-3.02	+658.65	+2.74
Test 4, 20kbps stereo	128.00	-679.69	-3.40	+695.31	+3.48
Test 5, 16kbps stereo	71.11	-914.06	-5.71	+703.13	+4.39
Test 6, 24kbps mono	119.94	-917.1	-3.82	+917.1	+3.82
Test 7, 20kbps mono	128.00	-593.75	-2.97	+554.69	+2.77
Test 8, 16kbps mono	71.11	-900	-5.62	+871.88	+5.45
Test 9, 12kbps mono	80.00	-750	-6.25	+550	+4.58

FIG 13

ori_pk_hash[]

arith_cf_ng_hash[]

1410

	index i (1420)	value[i+0] (1422)	value[i+1] (1424)	value[i+2] (1426)	value[i+3] (1428)	value[i+4] (1430)	value[i+5] (1432)	value[i+6] (1434)	value[i+7]
1412a	0	0x0000001E	0x147CBF16	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0x0D91390E
1412b	8	0x0E2EB915	0x0E4EB715	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0x17153315	0x0068F206	0x0DA0B104
1412c	16	0xFFFFFFFF	0x0D93AF0F	0x1009AF12	0x18932D0F	0xFFFFFFFF	0x10E72B10	0x1984EA15	0xFFFFFFFF
1412d	24	0xFFFFFFFF	0x0DD02711	0x0DA32701	0xFFFFFFFF	0xFFFFFFFF	0x12A22316	0xFFFFFFFF	0xFFFFFFFF
1412e	32	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0x145D1B16	0xFFFFFFFF	0xFFFFFFFF
1412f	40	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0x0E0F1513	0xFFFFFFFF	0x0E2F1314	0x0D91930E	0x0DA81304
1412g	48	0xFFFFFFFF	0x16F58F13	0x0422CE0B	0xFFFFFFFF	0x0422CC0A	0x0FEA0B13	0x19364A15	0xFFFFFFFF
1412h	56	0x0422C809	0x14BC0716	0xFFFFFFFF	0x0D90850D	0xFFFFFFFF	0x0DB08310	0x1893030F	0x00000103
1412i	64	0xFFFFFFFF	0x00097F00	0x0E8DFF17	0xFFFFFFFF	0x0068BC05	0x0D92FB1A	0x135FFB11	0x0025F901
1412j	72	0x0D9C7901	0x002F7702	0x1048F715	0x0038F503	0x0DAF7501	0x00427304	0x1893320F	0xFFFFFFFF
1412k	80	0xFFFFFFFF	0x0E0F6F13	0xFFFFFFFF	0x0D91ED1A	0xFFFFFFFF	0xFFFFFFFF	0x1980AA15	0xFFFFFFFF
1412l	88	0x00692807	0x13BEE711	0x17B36711	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
1412m	96	0xFFFFFFFF	0x0D90DF0D	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0x0E6E5B16	0x1262DB16	0x16575B11
1412n	104	0xFFFFFFFF	0x0068D705	0x0F4BD716	0x0D93550E	0xFFFFFFFF	0x10295314	0xFFFFFFFF	0xFFFFFFFF
1412o	112	0xFFFFFFFF	0x0422CF0C	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0x0DEFCB12	0x18940A04	0xFFFFFFFF
1412p	120	0x005F0800	0x12C1C716	0x1893080F	0xFFFFFFFF	0xFFFFFFFF	0x00694308	0x1893020F	0xFFFFFFFF

FIG 14

1510 arith_cf_ne[]

1520 1522 1524 1526 1528 1530 1532 1534

1512a 1512b

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]
0	0	32768	0	43690	21845	0	49152	32768
8	16384	0	52428	39321	26214	13107	0	54613
16	43690	32768	21845	10922	0	56173	46811	37449
24	28086	18724	9362	0	57344	49152	40960	32768
32	24576	16384	8192	0	58254	50972	43690	36408
40	29127	21845	14563	7281	0	58982	52428	45875
48	39321	32768	26214	19660	13107	6553	0	59578
56	53620	47662	41704	35746	29789	23831	17873	11915
64	5957	0	60074	54613	49152	43690	38229	32768
72	27306	21845	16384	10922	5461	0	60494	55453
80	50412	45371	40329	35288	30247	25206	20164	15123
88	10082	5041	0	60854	56173	51492	46811	42130
96	37449	32768	28086	23405	18724	14043	9362	4681
104	0	61166	56797	52428	48059	43690	39321	34952
112	30583	26214	21845	17476	13107	8738	4369	0
120	61440	57344	53248	49152	45056	40960	36864	32768
128	28672	24576	20480	16384	12288	8192	4096	0
136	61680	57825	53970	50115	46260	42405	38550	34695
144	30840	26985	23130	19275	15420	11565	7710	3855
152	0	61895	58254	54613	50972	47331	43690	40049
160	36408	32768	29127	25486	21845	18204	14563	10922
168	7281	3640	0	62086	58637	55188	51738	48289
176	44840	41391	37941	34492	31043	27594	24144	20695
184	17246	13797	10347	6898	3449	0	62259	58982
192	55705	52428	49152	45875	42598	39321	36044	32768
200	29491	26214	22937	19660	16384	13107	9830	6553
208	3276	0	62415	59294	56173	53052	49932	46811
216	43690	40569	37449	34328	31207	28086	24966	21845
224	18724	15603	12483	9362	6241	3120	0	62557
232	59578	56599	53620	50641	47662	44683	41704	38725
240	35746	32768	29789	26810	23831	20852	17873	14894

FIG 15(1)

248	11915	8936	5957	2978	0	62686	59837	56987
256	54138	51289	48439	45590	42740	39891	37042	34192
264	31343	28493	25644	22795	19945	17096	14246	11397
272	8548	5698	2849	0	62805	60074	57344	54613
280	51882	49152	46421	43690	40960	38229	35498	32768
288	30037	27306	24576	21845	19114	16384	13653	10922
296	8192	5461	2730	0	62914	60293	57671	55050
304	52428	49807	47185	44564	41943	39321	36700	34078
312	31457	28835	26214	23592	20971	18350	15728	13107
320	10485	7864	5242	2621	0	63015	60494	57974
328	55453	52932	50412	47891	45371	42850	40329	37809
336	35288	32768	30247	27726	25206	22685	20164	17644
344	15123	12603	10082	7561	5041	2520	0	63108
352	60681	58254	55826	53399	50972	48545	46117	43690
360	41263	38836	36408	33981	31554	29127	26699	24272
368	21845	19418	16990	14563	12136	9709	7281	4854
376	2427	0	63195	60854	58514	56173	53833	51492
384	49152	46811	44470	42130	39789	37449	35108	32768
392	30427	28086	25746	23405	21065	18724	16384	14043
400	11702	9362	7021	4681	2340	0	63276	61016
408	58756	56496	54236	51976	49716	47457	45197	42937
416	40677	38417	36157	33897	31638	29378	27118	24858
424	22598	20338	18078	15819	13559	11299	9039	6779
432	4519	2259	0	63351	61166	58982	56797	54613
440	52428	50244	48059	45875	43690	41506	39321	37137
448	34952	32768	30583	28398	26214	24029	21845	19660
456	17476	15291	13107	10922	8738	6553	4369	2184
464	0	63421	61307	59193	57079	54965	52851	50737
472	48623	46509	44395	42281	40167	38053	35939	33825
480	31710	29596	27482	25368	23254	21140	19026	16912
488	14798	12684	10570	8456	6342	4228	2114	0
496	63488	61440	59392	57344	55296	53248	51200	49152
504	47104	45056	43008	40960	38912	36864	34816	32768
512	30720	28672	26624	24576	22528	20480	18432	16384
520	14336	12288	10240	8192	6144	4096	2048	0
528	63550	61564	59578	57592	55606	53620	51634	49648

FIG 15(2)

536	47662	45676	43690	41704	39718	37732	35746	33760
544	31775	29789	27803	25817	23831	21845	19859	17873
552	15887	13901	11915	9929	7943	5957	3971	1985
560	0	63608	61680	59753	57825	55898	53970	52043
568	50115	48188	46260	44333	42405	40478	38550	36623
576	34695	32768	30840	28912	26985	25057	23130	21202
584	19275	17347	15420	13492	11565	9637	7710	5782
592	3855	1927	0	63663	61791	59918	58046	56173
600	54301	52428	50556	48683	46811	44938	43066	41194
608	39321	37449	35576	33704	31831	29959	28086	26214
616	24341	22469	20597	18724	16852	14979	13107	11234
624	9362	7489	5617	3744	1872	0	63715	61895
632	60074	58254	56433	54613	52792	50972	49152	47331
640	45511	43690	41870	40049	38229	36408	34588	32768
648	30947	29127	27306	25486	23665	21845	20024	18204
656	16384	14563	12743	10922	9102	7281	5461	3640
664	1820	0	63764	61993	60222	58451	56679	54908
672	53137	51366	49594	47823	46052	44281	42509	40738
680	38967	37196	35424	33653	31882	30111	28339	26568
688	24797	23026	21254	19483	17712	15941	14169	12398
696	10627	8856	7084	5313	3542	1771	0	63811
704	62086	60362	58637	56912	55188	53463	51738	50014
712	48289	46565	44840	43115	41391	39666	37941	36217
720	34492	32768	31043	29318	27594	25869	24144	22420
728	20695	18970	17246	15521	13797	12072	10347	8623
736	6898	5173	3449	1724	0	63855	62175	60494
744	58814	57133	55453	53773	52092	50412	48731	47051
752	45371	43690	42010	40329	38649	36969	35288	33608
760	31927	30247	28566	26886	25206	23525	21845	20164
768	18484	16804	15123	13443	11762	10082	8402	6721
776	5041	3360	1680	0	63897	62259	60620	58982
784	57344	55705	54067	52428	50790	49152	47513	45875
792	44236	42598	40960	39321	37683	36044	34406	32768
800	31129	29491	27852	26214	24576	22937	21299	19660
808	18022	16384	14745	13107	11468	9830	8192	6553
816	4915	3276	1638	0	63937	62339	60740	59142

FIG 15(3)

824	57543	55945	54346	52748	51150	49551	47953	46354
832	44756	43157	41559	39960	38362	36764	35165	33567
840	31968	30370	28771	27173	25575	23976	22378	20779
848	19181	17582	15984	14385	12787	11189	9590	7992
856	6393	4795	3196	1598	0	63975	62415	60854
864	59294	57734	56173	54613	53052	51492	49932	48371
872	46811	45251	43690	42130	40569	39009	37449	35888
880	34328	32768	31207	29647	28086	26526	24966	23405
888	21845	20284	18724	17164	15603	14043	12483	10922
896	9362	7801	6241	4681	3120	1560	0	64011
904	62487	60963	59439	57915	56391	54867	53343	51819
912	50295	48770	47246	45722	44198	42674	41150	39626
920	38102	36578	35054	33530	32005	30481	28957	27433
928	25909	24385	22861	21337	19813	18289	16765	15240
936	13716	12192	10668	9144	7620	6096	4572	3048
944	1524	0	64046	62557	61067	59578	58088	56599
952	55109	53620	52130	50641	49152	47662	46173	44683
960	43194	41704	40215	38725	37236	35746	34257	32768
968	31278	29789	28299	26810	25320	23831	22341	20852
976	19362	17873	16384	14894	13405	11915	10426	8936
984	7447	5957	4468	2978	1489	0	64079	62623
992	61166	59710	58254	56797	55341	53885	52428	50972
1000	49516	48059	46603	45147	43690	42234	40777	39321
1008	37865	36408	34952	33496	32039	30583	29127	27670
1016	26214	24758	23301	21845	20388	18932	17476	16019
1024	14563	13107	11650	10194	8738	7281	5825	4369
1032	2912	1456	0	64111	62686	61261	59837	58412
1040	56987	55563	54138	52713	51289	49864	48439	47014
1048	45590	44165	42740	41316	39891	38466	37042	35617
1056	34192	32768	31343	29918	28493	27069	25644	24219
1064	22795	21370	19945	18521	17096	15671	14246	12822
1072	11397	9972	8548	7123	5698	4274	2849	1424
1080	0	64141	62747	61352	59958	58564	57169	55775
1088	54380	52986	51592	50197	48803	47409	46014	44620
1096	43225	41831	40437	39042	37648	36253	34859	33465
1104	32070	30676	29282	27887	26493	25098	23704	22310

FIG 15(4)

1112	20915	19521	18126	16732	15338	13943	12549	11155
1120	9760	8366	6971	5577	4183	2788	1394	0
1128	64170	62805	61440	60074	58709	57344	55978	54613
1136	53248	51882	50517	49152	47786	46421	45056	43690
1144	42325	40960	39594	38229	36864	35498	34133	32768
1152	31402	30037	28672	27306	25941	24576	23210	21845
1160	20480	19114	17749	16384	15018	13653	12288	10922
1168	9557	8192	6826	5461	4096	2730	1365	0
1176	64198	62861	61523	60186	58848	57511	56173	54836
1184	53498	52161	50823	49486	48148	46811	45473	44136
1192	42799	41461	40124	38786	37449	36111	34774	33436
1200	32099	30761	29424	28086	26749	25411	24074	22736
1208	21399	20062	18724	17387	16049	14712	13374	12037
1216	10699	9362	8024	6687	5349	4012	2674	1337
1224	0	64225	62914	61603	60293	58982	57671	56360
1232	55050	53739	52428	51118	49807	48496	47185	45875
1240	44564	43253	41943	40632	39321	38010	36700	35389
1248	34078	32768	31457	30146	28835	27525	26214	24903
1256	23592	22282	20971	19660	18350	17039	15728	14417
1264	13107	11796	10485	9175	7864	6553	5242	3932
1272	2621	1310	0	64250	62965	61680	60395	59110
1280	57825	56540	55255	53970	52685	51400	50115	48830
1288	47545	46260	44975	43690	42405	41120	39835	38550
1296	37265	35980	34695	33410	32125	30840	29555	28270
1304	26985	25700	24415	23130	21845	20560	19275	17990
1312	16705	15420	14135	12850	11565	10280	8995	7710
1320	6425	5140	3855	2570	1285	0	64275	63015
1328	61755	60494	59234	57974	56713	55453	54193	52932
1336	51672	50412	49152	47891	46631	45371	44110	42850
1344	41590	40329	39069	37809	36548	35288	34028	32768
1352	31507	30247	28987	27726	26466	25206	23945	22685
1360	21425	20164	18904	17644	16384	15123	13863	12603
1368	11342	10082	8822	7561	6301	5041	3780	2520
1376	1260	0	64299	63062	61826	60589	59353	58116
1384	56880	55643	54407	53170	51934	50697	49461	48224
1392	46988	45751	44515	43278	42041	40805	39568	38332

FIG 15(5)

1400	37095	35859	34622	33386	32149	30913	29676	28440
1408	27203	25967	24730	23494	22257	21020	19784	18547
1416	17311	16074	14838	13601	12365	11128	9892	8655
1424	7419	6182	4946	3709	2473	1236	0	64322
1432	63108	61895	60681	59467	58254	57040	55826	54613
1440	53399	52186	50972	49758	48545	47331	46117	44904
1448	43690	42477	41263	40049	38836	37622	36408	35195
1456	33981	32768	31554	30340	29127	27913	26699	25486
1464	24272	23058	21845	20631	19418	18204	16990	15777
1472	14563	13349	12136	10922	9709	8495	7281	6068
1480	4854	3640	2427	1213	0	64344	63152	61961
1488	60769	59578	58386	57195	56003	54811	53620	52428
1496	51237	50045	48854	47662	46470	45279	44087	42896
1504	41704	40513	39321	38130	36938	35746	34555	33363
1512	32172	30980	29789	28597	27405	26214	25022	23831
1520	22639	21448	20256	19065	17873	16681	15490	14298
1528	13107	11915	10724	9532	8340	7149	5957	4766
1536	3574	2383	1191	0	64365	63195	62025	60854
1544	59684	58514	57344	56173	55003	53833	52662	51492
1552	50322	49152	47981	46811	45641	44470	43300	42130
1560	40960	39789	38619	37449	36278	35108	33938	32768
1568	31597	30427	29257	28086	26916	25746	24576	23405
1576	22235	21065	19894	18724	17554	16384	15213	14043
1584	12873	11702	10532	9362	8192	7021	5851	4681
1592	3510	2340	1170	0	64386	63236	62086	60936
1600	59787	58637	57487	56337	55188	54038	52888	51738
1608	50589	49439	48289	47139	45990	44840	43690	42540
1616	41391	40241	39091	37941	36792	35642	34492	33342
1624	32193	31043	29893	28743	27594	26444	25294	24144
1632	22995	21845	20695	19545	18396	17246	16096	14946
1640	13797	12647	11497	10347	9198	8048	6898	5748
1648	4599	3449	2299	1149	0	64406	63276	62146
1656	61016	59886	58756	57626	56496	55366	54236	53106
1664	51976	50846	49716	48587	47457	46327	45197	44067
1672	42937	41807	40677	39547	38417	37287	36157	35027
1680	33897	32768	31638	30508	29378	28248	27118	25988

FIG 15(6)

1688	24858	23728	22598	21468	20338	19208	18078	16948
1696	15819	14689	13559	12429	11299	10169	9039	7909
1704	6779	5649	4519	3389	2259	1129	0	64425
1712	63314	62203	61092	59982	58871	57760	56649	55538
1720	54428	53317	52206	51095	49985	48874	47763	46652
1728	45541	44431	43320	42209	41098	39988	38877	37766
1736	36655	35544	34434	33323	32212	31101	29991	28880
1744	27769	26658	25547	24437	23326	22215	21104	19994
1752	18883	17772	16661	15550	14440	13329	12218	11107
1760	9997	8886	7775	6664	5553	4443	3332	2221
1768	1110	0	64443	63351	62259	61166	60074	58982
1776	57890	56797	55705	54613	53521	52428	51336	50244
1784	49152	48059	46967	45875	44782	43690	42598	41506
1792	40413	39321	38229	37137	36044	34952	33860	32768
1800	31675	30583	29491	28398	27306	26214	25122	24029
1808	22937	21845	20753	19660	18568	17476	16384	15291
1816	14199	13107	12014	10922	9830	8738	7645	6553
1824	5461	4369	3276	2184	1092	0	64461	63387
1832	62312	61238	60164	59089	58015	56941	55866	54792
1840	53718	52643	51569	50494	49420	48346	47271	46197
1848	45123	44048	42974	41900	40825	39751	38676	37602
1856	36528	35453	34379	33305	32230	31156	30082	29007
1864	27933	26859	25784	24710	23635	22561	21487	20412
1872	19338	18264	17189	16115	15041	13966	12892	11817
1880	10743	9669	8594	7520	6446	5371	4297	3223
1888	2148	1074	0	64478	63421	62364	61307	60250
1896	59193	58136	57079	56022	54965	53908	52851	51794
1904	50737	49680	48623	47566	46509	45452	44395	43338
1912	42281	41224	40167	39110	38053	36996	35939	34882
1920	33825	32768	31710	30653	29596	28539	27482	26425
1928	25368	24311	23254	22197	21140	20083	19026	17969
1936	16912	15855	14798	13741	12684	11627	10570	9513
1944	8456	7399	6342	5285	4228	3171	2114	1057
1952	0	64495	63455	62415	61374	60334	59294	58254
1960	57213	56173	55133	54093	53052	52012	50972	49932
1968	48891	47851	46811	45771	44730	43690	42650	41610

FIG 15(7)

1976	40569	39529	38489	37449	36408	35368	34328	33288
1984	32247	31207	30167	29127	28086	27046	26006	24966
1992	23925	22885	21845	20805	19764	18724	17684	16644
2000	15603	14563	13523	12483	11442	10402	9362	8322
2008	7281	6241	5201	4161	3120	2080	1040	0
2016	64512	63488	62464	61440	60416	59392	58368	57344
2024	56320	55296	54272	53248	52224	51200	50176	49152
2032	48128	47104	46080	45056	44032	43008	41984	40960
2040	39936	38912	37888	36864	35840	34816	33792	32768
2048	31744	30720	29696	28672	27648	26624	25600	24576
2056	23552	22528	21504	20480	19456	18432	17408	16384
2064	15360	14336	13312	12288	11264	10240	9216	8192
2072	7168	6144	5120	4096	3072	2048	1024	0
2080	64527	63519	62511	61503	60494	59486	58478	57470
2088	56461	55453	54445	53437	52428	51420	50412	49404
2096	48395	47387	46379	45371	44362	43354	42346	41338
2104	40329	39321	38313	37305	36296	35288	34280	33272
2112	32263	31255	30247	29239	28230	27222	26214	25206
2120	24197	23189	22181	21173	20164	19156	18148	17140
2128	16131	15123	14115	13107	12098	11090	10082	9074
2136	8065	7057	6049	5041	4032	3024	2016	1008
2144	0	64543	63550	62557	61564	60571	59578	58585
2152	57592	56599	55606	54613	53620	52627	51634	50641
2160	49648	48655	47662	46669	45676	44683	43690	42697
2168	41704	40711	39718	38725	37732	36739	35746	34753
2176	33760	32768	31775	30782	29789	28796	27803	26810
2184	25817	24824	23831	22838	21845	20852	19859	18866
2192	17873	16880	15887	14894	13901	12908	11915	10922
2200	9929	8936	7943	6950	5957	4964	3971	2978
2208	1985	992	0	64557	63579	62601	61623	60645
2216	59667	58688	57710	56732	55754	54776	53798	52820
2224	51841	50863	49885	48907	47929	46951	45973	44994
2232	44016	43038	42060	41082	40104	39125	38147	37169
2240	36191	35213	34235	33257	32278	31300	30322	29344
2248	28366	27388	26410	25431	24453	23475	22497	21519
2256	20541	19562	18584	17606	16628	15650	14672	13694

FIG 15(8)

2264	12715	11737	10759	9781	8803	7825	6847	5868
2272	4890	3912	2934	1956	978	0	64572	63608
2280	62644	61680	60717	59753	58789	57825	56862	55898
2288	54934	53970	53007	52043	51079	50115	49152	48188
2296	47224	46260	45296	44333	43369	42405	41441	40478
2304	39514	38550	37586	36623	35659	34695	33731	32768
2312	31804	30840	29876	28912	27949	26985	26021	25057
2320	24094	23130	22166	21202	20239	19275	18311	17347
2328	16384	15420	14456	13492	12528	11565	10601	9637
2336	8673	7710	6746	5782	4818	3855	2891	1927
2344	963	0	64586	63636	62686	61736	60787	59837
2352	58887	57937	56987	56038	55088	54138	53188	52238
2360	51289	50339	49389	48439	47489	46540	45590	44640
2368	43690	42740	41791	40841	39891	38941	37991	37042
2376	36092	35142	34192	33242	32293	31343	30393	29443
2384	28493	27544	26594	25644	24694	23744	22795	21845
2392	20895	19945	18995	18046	17096	16146	15196	14246
2400	13297	12347	11397	10447	9497	8548	7598	6648
2408	5698	4748	3799	2849	1899	949	0	64599
2416	63663	62727	61791	60854	59918	58982	58046	57109
2424	56173	55237	54301	53365	52428	51492	50556	49620
2432	48683	47747	46811	45875	44938	44002	43066	42130
2440	41194	40257	39321	38385	37449	36512	35576	34640
2448	33704	32768	31831	30895	29959	29023	28086	27150
2456	26214	25278	24341	23405	22469	21533	20597	19660
2464	18724	17788	16852	15915	14979	14043	13107	12170
2472	11234	10298	9362	8426	7489	6553	5617	4681
2480	3744	2808	1872	936	0	64612	63689	62766
2488	61843	60920	59997	59074	58151	57228	56305	55382
2496	54459	53536	52613	51690	50767	49844	48921	47998
2504	47075	46152	45229	44306	43382	42459	41536	40613
2512	39690	38767	37844	36921	35998	35075	34152	33229
2520	32306	31383	30460	29537	28614	27691	26768	25845
2528	24922	23999	23076	22153	21229	20306	19383	18460
2536	17537	16614	15691	14768	13845	12922	11999	11076
2544	10153	9230	8307	7384	6461	5538	4615	3692

FIG 15(9)

2552	2769	1846	923	0	64625	63715	62805	61895
2560	60984	60074	59164	58254	57344	56433	55523	54613
2568	53703	52792	51882	50972	50062	49152	48241	47331
2576	46421	45511	44600	43690	42780	41870	40960	40049
2584	39139	38229	37319	36408	35498	34588	33678	32768
2592	31857	30947	30037	29127	28216	27306	26396	25486
2600	24576	23665	22755	21845	20935	20024	19114	18204
2608	17294	16384	15473	14563	13653	12743	11832	10922
2616	10012	9102	8192	7281	6371	5461	4551	3640
2624	2730	1820	910	0	64638	63740	62842	61944
2632	61047	60149	59251	58353	57456	56558	55660	54762
2640	53865	52967	52069	51171	50274	49376	48478	47580
2648	46683	45785	44887	43989	43092	42194	41296	40398
2656	39501	38603	37705	36807	35910	35012	34114	33216
2664	32319	31421	30523	29625	28728	27830	26932	26034
2672	25137	24239	23341	22443	21546	20648	19750	18852
2680	17955	17057	16159	15261	14364	13466	12568	11670
2688	10773	9875	8977	8079	7182	6284	5386	4488
2696	3591	2693	1795	897	0			

FIG 15(10)

arith_cf_ng[pki][545]

1640 pki=0

1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665

1642a

1642b

1601

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	16684	15615	14544	13389	12025	10634	9484	8407	7344	7199	7072	6949	6820	6686	6509	6352
16	6202	6049	5857	5635	5375	5113	4893	4699	4544	4394	4235	4061	3924	3795	3671	3547
32	3405	3366	3337	3305	3270	3236	3198	3160	3126	3076	3040	3006	2972	2909	2865	2789
48	2745	2698	2622	2575	2514	2480	2445	2408	2353	2318	2281	2244	2210	2178	2146	2117
64	2075	2066	2038	2025	2013	1999	1987	1954	1940	1928	1919	1905	1873	1840	1827	1819
80	1807	1792	1763	1749	1736	1723	1710	1684	1675	1670	1667	1665	1663	1660	1655	1650
96	1646	1642	1637	1633	1627	1623	1619	1615	1610	1603	1594	1586	1579	1569	1558	1538
112	1515	1492	1472	1463	1454	1446	1438	1428	1421	1416	1412	1409	1404	1399	1394	1389
128	1384	1380	1375	1370	1367	1364	1362	1359	1355	1353	1351	1349	1346	1344	1343	1342
144	1340	1338	1336	1334	1331	1328	1326	1323	1320	1317	1314	1311	1308	1305	1302	1298
160	1294	1290	1287	1284	1281	1277	1274	1271	1268	1264	1262	1260	1256	1254	1251	1248
176	1245	1240	1233	1224	1217	1208	1201	1194	1187	1180	1174	1168	1158	1152	1144	1137
192	1132	1128	1126	1122	1120	1116	1113	1111	1107	1104	1101	1098	1095	1093	1090	1087
208	1083	1078	1075	1071	1068	1064	1061	1059	1056	1052	1050	1047	1044	1041	1039	1037
224	1035	1033	1031	1030	1028	1026	1024	1022	1020	1019	1018	1016	1015	1014	1013	1012
240	1010	1008	1006	1004	1002	999	997	994	992	990	988	985	983	980	977	975
256	971	968	966	964	962	960	957	955	953	951	949	946	944	942	939	937
272	934	930	927	923	920	918	916	914	912	910	907	904	901	899	898	896
288	894	893	892	890	889	888	887	886	884	883	882	881	880	879	878	877
304	876	874	872	868	864	859	855	853	852	851	849	848	847	846	845	844
320	843	839	832	825	820	811	804	797	793	785	778	772	769	762	755	746
336	742	731	726	721	717	708	698	694	689	678	672	663	659	650	644	639
352	632	630	626	622	619	617	611	606	603	600	595	589	586	583	578	572
368	569	567	565	562	560	558	555	553	551	549	546	544	542	540	538	535
384	533	529	527	524	523	519	517	515	514	510	508	506	505	502	500	498
400	496	494	490	486	484	482	478	475	473	470	466	462	460	458	454	450
416	447	444	442	440	438	436	433	431	428	425	422	419	417	415	412	410
432	408	407	406	404	403	402	401	400	399	397	395	391	390	387	386	384
448	383	381	379	377	375	373	370	368	365	363	361	357	355	353	349	345
464	342	341	340	338	336	335	333	331	329	327	325	321	319	316	313	310
480	307	306	305	304	303	302	300	299	297	295	294	292	291	289	287	285
496	283	282	281	280	279	278	277	276	275	274	273	271	270	269	267	265
512	263	262	261	260	259	258	257	256	255	254	253	252	251	250	249	248
528	247	246	245	244	243	242	241	240	239	238	237	236	235	234	233	232
544	0															

FIG 16(1)

1602

pki = 1

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	42136	39764	37612	35527	33187	30758	28572	26377	23758	23277	22819	22398	21959	21551	21004	20564
16	20153	19671	19128	18629	18038	17406	16913	16401	15967	15495	15047	14489	13968	13514	13041	12599
32	12109	11928	11762	11631	11467	11316	11163	11000	10866	10672	10522	10343	10203	9993	9842	9603
48	9459	9326	9098	8948	8762	8607	8460	8266	8058	7883	7705	7555	7412	7271	7086	6952
64	6771	6731	6610	6543	6461	6397	6339	6205	6157	6094	6067	6038	5982	5913	5885	5855
80	5802	5714	5605	5532	5442	5378	5307	5217	5165	5138	5117	5101	5090	5075	5043	5024
96	5003	4988	4966	4944	4922	4905	4887	4875	4856	4840	4820	4799	4781	4766	4744	4708
112	4666	4629	4588	4561	4539	4523	4500	4475	4460	4441	4425	4414	4398	4373	4351	4330
128	4311	4296	4279	4254	4232	4216	4197	4182	4160	4149	4140	4131	4121	4115	4109	4105
144	4095	4079	4072	4063	4050	4040	4030	4019	4003	3986	3976	3962	3950	3942	3926	3915
160	3893	3879	3871	3860	3847	3831	3820	3809	3798	3782	3768	3755	3742	3735	3723	3712
176	3700	3689	3679	3666	3655	3630	3615	3595	3571	3565	3543	3526	3501	3485	3463	3452
192	3441	3435	3426	3412	3404	3394	3388	3374	3363	3352	3339	3325	3318	3307	3298	3283
208	3272	3256	3246	3230	3224	3211	3192	3178	3167	3159	3148	3136	3125	3114	3108	3090
224	3077	3066	3058	3048	3044	3029	3023	3014	3003	2996	2990	2985	2979	2969	2962	2951
240	2940	2935	2925	2920	2914	2903	2898	2881	2873	2862	2851	2842	2835	2824	2811	2800
256	2778	2769	2756	2748	2740	2733	2728	2718	2710	2699	2691	2674	2663	2656	2646	2643
272	2632	2613	2603	2592	2589	2580	2563	2550	2542	2531	2522	2513	2507	2501	2495	2490
288	2477	2469	2457	2451	2443	2434	2426	2420	2413	2407	2405	2401	2400	2393	2384	2378
304	2375	2374	2368	2363	2349	2343	2340	2336	2331	2329	2324	2318	2314	2313	2310	2307
320	2305	2285	2266	2244	2223	2208	2188	2171	2155	2139	2114	2089	2078	2049	2018	2000
336	1978	1946	1929	1908	1892	1862	1830	1814	1802	1769	1746	1721	1703	1683	1658	1642
352	1621	1618	1607	1591	1585	1579	1563	1554	1545	1537	1531	1524	1516	1512	1505	1492
368	1478	1473	1465	1459	1450	1448	1438	1430	1424	1421	1412	1402	1399	1390	1380	1375
384	1368	1359	1355	1350	1348	1340	1329	1323	1317	1307	1298	1288	1279	1273	1267	1263
400	1256	1252	1239	1227	1221	1215	1199	1192	1185	1181	1173	1166	1160	1153	1144	1140
416	1134	1127	1121	1115	1107	1100	1089	1082	1076	1071	1066	1058	1054	1051	1044	1041
432	1032	1031	1028	1027	1023	1019	1016	1010	1008	1002	1001	995	994	989	984	979
448	973	970	959	953	946	939	929	926	920	914	905	897	881	878	867	856
464	847	844	839	835	830	829	823	817	816	811	807	803	787	782	776	772
480	762	757	753	749	743	742	737	733	727	723	720	716	712	711	705	699
496	689	688	685	684	680	678	670	668	663	662	656	652	647	640	638	636
512	633	632	630	629	627	626	624	623	622	621	620	619	618	617	616	613
528	612	611	610	609	608	607	606	605	604	603	600	599	598	597	596	595
544	0															

FIG 16(2)

1603

pki=2

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	45864	43310	41135	39195	37170	35025	33027	30777	27617	26936	26531	26103	25632	25199	24626	24176
16	23709	23296	22794	22325	21785	21258	20787	20302	19774	19364	18895	18078	17543	17051	16525	15992
32	15418	15118	14946	14744	14593	14444	14305	14176	13979	13813	13670	13537	13343	13126	12981	12745
48	12582	12418	12210	12051	11881	11728	11528	11303	10937	10799	10560	10351	10220	10020	9867	9728
64	9590	9524	9250	9173	9107	9028	8978	8904	8825	8763	8731	8701	8648	8596	8567	8517
80	8462	8357	8055	7973	7889	7796	7738	7678	7591	7565	7545	7528	7517	7508	7444	7413
96	7396	7371	7357	7341	7312	7290	7274	7254	7232	7215	7187	7157	7126	7101	7076	7041
112	6990	6945	6911	6887	6872	6855	6832	6818	6796	6764	6749	6734	6711	6685	6664	6640
128	6617	6599	6571	6516	6496	6480	6457	6435	6393	6377	6370	6358	6344	6332	6323	6317
144	6307	6287	6273	6261	6244	6224	6215	6202	6188	6178	6166	6151	6137	6122	6109	6086
160	6068	6053	6030	6018	6000	5988	5976	5964	5945	5925	5909	5895	5875	5861	5847	5835
176	5818	5801	5783	5760	5736	5704	5684	5659	5636	5622	5598	5574	5554	5533	5510	5497
192	5480	5466	5451	5437	5426	5403	5389	5375	5352	5332	5318	5291	5261	5251	5233	5221
208	5201	5181	5176	5159	5151	5134	5123	5108	5095	5077	5061	5047	5033	5010	4997	4977
224	4952	4940	4926	4912	4896	4871	4862	4851	4826	4810	4802	4792	4781	4772	4761	4749
240	4743	4726	4717	4708	4696	4682	4674	4652	4634	4628	4613	4599	4590	4582	4568	4556
256	4544	4532	4523	4510	4492	4481	4467	4455	4442	4429	4417	4400	4383	4377	4367	4356
272	4338	4306	4292	4261	4244	4230	4211	4202	4190	4179	4165	4159	4147	4133	4124	4111
288	4099	4090	4078	4064	4052	4046	4036	4030	4019	4015	4012	4011	4009	4006	3989	3985
304	3980	3974	3959	3945	3939	3930	3921	3912	3909	3904	3901	3892	3890	3884	3881	3880
320	3866	3834	3807	3765	3740	3709	3686	3647	3629	3609	3592	3558	3541	3499	3459	3414
336	3379	3353	3327	3298	3278	3245	3214	3182	3153	3108	3080	3005	2984	2953	2918	2885
352	2829	2818	2804	2788	2778	2766	2750	2730	2716	2703	2689	2666	2653	2641	2628	2610
368	2600	2591	2580	2566	2554	2542	2533	2519	2509	2502	2493	2479	2467	2460	2453	2444
384	2429	2418	2411	2393	2380	2368	2359	2350	2347	2333	2321	2291	2282	2267	2252	2243
400	2223	2215	2205	2189	2173	2164	2156	2135	2131	2119	2111	2084	2076	2064	2050	2030
416	2019	2010	1996	1990	1974	1961	1947	1934	1926	1920	1913	1896	1886	1879	1870	1858
432	1838	1835	1831	1814	1808	1804	1801	1798	1795	1789	1784	1770	1767	1763	1759	1747
448	1730	1722	1715	1698	1685	1677	1668	1647	1636	1630	1619	1592	1580	1571	1561	1537
464	1512	1506	1501	1486	1476	1467	1461	1441	1427	1423	1417	1400	1394	1385	1374	1356
480	1340	1337	1332	1320	1311	1305	1303	1294	1286	1277	1273	1266	1258	1254	1251	1239
496	1227	1226	1225	1220	1217	1214	1211	1206	1199	1195	1193	1185	1179	1177	1173	1158
512	1146	1145	1144	1141	1139	1138	1136	1130	1128	1126	1125	1123	1120	1118	1117	1111
528	1106	1105	1104	1103	1102	1101	1098	1097	1096	1095	1094	1093	1092	1091	1088	1085
544	0															

FIG 16(3)

pki=3

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	45913	42800	40542	38545	36406	34310	32344	30248	27777	27220	26674	26155	25659	25139	24316	23859
16	23474	22952	22390	21868	21323	20784	20350	19871	19461	18955	18491	17934	17484	17037	16634	16209
32	15553	15409	15269	15127	14903	14771	14552	14319	14181	13787	13556	13344	13179	13001	12821	12612
48	12446	12302	12077	11935	11740	11574	11432	11291	11103	10890	10774	10646	10483	10341	10142	9973
64	9666	9584	9519	9448	9373	9290	9191	8917	8823	8756	8706	8670	8633	8574	8549	8511
80	8434	8361	8281	8223	8143	8083	8008	7741	7676	7615	7590	7572	7555	7536	7490	7449
96	7436	7403	7386	7366	7338	7312	7297	7278	7245	7231	7205	7184	7159	7138	7104	7063
112	7026	6983	6945	6920	6902	6878	6848	6815	6793	6771	6760	6741	6717	6676	6657	6646
128	6625	6614	6591	6535	6521	6507	6490	6468	6439	6417	6399	6390	6376	6370	6357	6340
144	6326	6296	6276	6262	6243	6226	6207	6196	6179	6167	6159	6148	6133	6126	6114	6100
160	6077	6052	6036	6018	6005	5982	5962	5944	5930	5902	5883	5875	5861	5850	5836	5814
176	5797	5779	5768	5744	5727	5713	5696	5678	5658	5642	5618	5605	5572	5552	5532	5515
192	5498	5478	5463	5448	5431	5412	5401	5382	5365	5343	5336	5323	5311	5292	5277	5254
208	5244	5227	5216	5200	5191	5173	5151	5138	5124	5107	5090	5076	5056	5034	5019	5012
224	4986	4979	4968	4960	4950	4930	4916	4906	4889	4880	4874	4863	4854	4844	4823	4814
240	4805	4790	4781	4767	4753	4744	4737	4724	4714	4705	4691	4672	4664	4642	4615	4601
256	4553	4534	4523	4511	4498	4480	4450	4439	4431	4420	4407	4393	4382	4368	4361	4345
272	4336	4330	4318	4306	4301	4290	4283	4262	4256	4248	4228	4217	4204	4195	4182	4171
288	4150	4144	4135	4122	4113	4104	4095	4089	4072	4057	4055	4051	4050	4044	4034	4030
304	4027	4022	4013	4002	3991	3985	3974	3960	3954	3951	3945	3927	3922	3920	3917	3916
320	3908	3878	3849	3814	3788	3769	3744	3719	3699	3670	3645	3622	3603	3568	3527	3496
336	3466	3414	3386	3364	3346	3318	3251	3238	3219	3171	3135	3094	3061	3031	2997	2969
352	2933	2923	2901	2886	2875	2864	2845	2836	2824	2818	2804	2788	2778	2769	2741	2731
368	2717	2704	2696	2682	2672	2659	2642	2635	2627	2618	2601	2590	2581	2569	2548	2535
384	2525	2494	2485	2481	2472	2463	2441	2435	2430	2421	2410	2398	2388	2372	2354	2346
400	2336	2328	2303	2294	2283	2272	2256	2246	2237	2228	2216	2207	2196	2183	2163	2150
416	2141	2127	2117	2106	2093	2082	2063	2056	2052	2044	2030	2020	2010	1996	1977	1960
432	1954	1946	1939	1933	1925	1923	1918	1912	1906	1899	1890	1885	1874	1867	1859	1854
448	1848	1837	1828	1818	1810	1800	1783	1777	1769	1752	1735	1721	1706	1696	1682	1663
464	1652	1645	1640	1634	1625	1618	1608	1602	1598	1589	1578	1569	1562	1555	1534	1524
480	1517	1500	1494	1491	1489	1482	1468	1462	1459	1455	1449	1443	1440	1436	1421	1416
496	1409	1406	1403	1398	1396	1390	1380	1376	1372	1368	1362	1359	1353	1345	1337	1335
512	1333	1332	1331	1330	1328	1327	1324	1323	1322	1321	1318	1317	1314	1311	1305	1302
528	1300	1299	1298	1297	1296	1295	1294	1293	1292	1291	1290	1289	1288	1286	1285	1284
544	0															

1604

FIG 16(4)

pki = 4

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	42444	39826	37593	35349	32952	30547	28391	26250	23878	23355	22930	22470	22042	21556	20952	20517
16	20051	19561	19036	18502	17932	17297	16853	16341	15868	15450	14985	14428	14005	13534	13134	12659
32	12199	12016	11887	11730	11565	11443	11294	11120	10955	10739	10511	10377	10213	10017	9847	9583
48	9430	9271	9017	8887	8709	8573	8398	8237	8018	7884	7740	7553	7423	7285	7156	7010
64	6831	6780	6701	6648	6588	6507	6432	6308	6235	6171	6137	6116	6057	6007	5974	5947
80	5891	5813	5697	5628	5557	5481	5426	5303	5260	5235	5224	5211	5197	5179	5148	5133
96	5121	5100	5082	5071	5056	5040	5029	5007	4990	4967	4946	4932	4914	4894	4862	4827
112	4790	4758	4722	4695	4677	4661	4638	4609	4593	4577	4564	4550	4535	4517	4501	4485
128	4470	4455	4443	4406	4385	4372	4356	4332	4308	4297	4287	4279	4267	4261	4255	4246
144	4229	4216	4205	4193	4179	4165	4159	4153	4143	4130	4119	4108	4096	4086	4078	4066
160	4050	4035	4022	4007	3993	3981	3970	3954	3941	3925	3912	3901	3889	3881	3873	3862
176	3855	3833	3820	3804	3782	3758	3737	3712	3703	3692	3678	3658	3634	3615	3593	3577
192	3563	3552	3539	3523	3510	3499	3482	3475	3456	3445	3433	3416	3407	3398	3385	3375
208	3364	3348	3328	3309	3292	3277	3268	3254	3243	3228	3221	3213	3203	3189	3179	3171
224	3158	3152	3143	3132	3120	3102	3093	3084	3071	3067	3061	3055	3043	3038	3032	3023
240	3015	3004	2998	2991	2979	2972	2966	2958	2950	2946	2933	2915	2903	2899	2889	2878
256	2859	2851	2845	2834	2821	2815	2809	2800	2795	2789	2780	2767	2752	2743	2734	2727
272	2718	2705	2695	2673	2661	2653	2642	2629	2619	2613	2601	2591	2580	2572	2566	2559
288	2550	2538	2528	2519	2512	2502	2495	2489	2476	2472	2471	2469	2468	2466	2460	2456
304	2455	2450	2446	2440	2434	2428	2422	2416	2413	2410	2406	2399	2396	2395	2393	2391
320	2387	2368	2346	2326	2310	2287	2267	2245	2233	2211	2180	2165	2148	2130	2094	2066
336	2046	2017	1998	1976	1955	1930	1911	1890	1874	1847	1825	1788	1774	1745	1729	1707
352	1685	1672	1662	1652	1648	1639	1629	1623	1612	1607	1603	1591	1585	1571	1554	1537
368	1528	1522	1520	1510	1504	1499	1492	1488	1483	1480	1474	1462	1459	1453	1448	1436
384	1431	1420	1414	1411	1406	1398	1390	1381	1375	1364	1357	1343	1340	1332	1327	1319
400	1312	1300	1289	1276	1270	1264	1258	1250	1245	1238	1227	1212	1206	1193	1180	1169
416	1164	1159	1152	1141	1134	1127	1121	1112	1104	1098	1093	1080	1077	1070	1062	1055
432	1046	1040	1039	1032	1030	1029	1026	1021	1020	1014	1008	999	998	993	990	985
448	982	973	967	960	952	943	937	934	931	924	918	906	903	897	889	881
464	871	868	863	860	857	852	846	840	835	828	822	816	812	802	797	787
480	783	781	780	778	775	774	770	765	763	760	758	750	747	744	740	732
496	726	725	724	722	721	719	717	714	712	709	708	704	703	701	700	694
512	692	691	690	689	688	687	686	685	684	683	682	681	680	679	675	673
528	670	669	668	667	666	665	664	662	661	660	658	657	656	655	654	653
544	0															

1605

FIG 16(5)

pki=5

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	6137	5619	5100	4568	3982	3394	2861	2345	1824	1778	1745	1711	1676	1635	1586	1538
16	1501	1456	1405	1346	1287	1229	1170	1119	1074	1037	989	940	900	865	830	796
32	750	739	734	727	721	715	709	702	695	687	680	675	669	654	647	637
48	630	623	613	606	591	585	580	574	565	558	551	546	540	534	527	522
64	511	510	503	501	500	498	497	493	491	489	488	487	485	483	482	481
80	479	477	473	472	470	469	467	460	459	458	457	456	455	454	453	452
96	451	450	449	448	447	446	445	444	443	442	441	440	439	438	437	435
112	434	433	431	430	429	428	427	426	425	424	423	422	421	420	419	418
128	417	416	415	414	413	412	411	410	409	408	407	406	405	404	403	402
144	401	400	399	398	397	396	395	394	393	392	391	390	389	388	387	386
160	385	384	383	382	381	380	379	378	377	376	375	374	373	372	371	370
176	369	368	367	366	365	364	363	362	361	360	359	358	357	356	355	354
192	353	352	351	350	349	348	347	346	345	344	343	342	341	340	339	338
208	337	336	335	334	333	332	331	330	329	328	327	326	325	324	323	322
224	321	320	319	318	317	316	315	314	313	312	311	310	309	308	307	306
240	305	304	303	302	301	300	299	298	297	296	295	294	293	292	291	290
256	289	288	287	286	285	284	283	282	281	280	279	278	277	276	275	274
272	273	272	271	270	269	268	267	266	265	264	263	262	261	260	259	258
288	257	256	255	254	253	252	251	250	249	248	247	246	245	244	243	242
304	241	240	239	238	237	236	235	234	233	232	231	230	229	228	227	226
320	225	224	223	222	221	220	219	218	217	216	215	214	213	212	211	210
336	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194
352	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178
368	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162
384	161	160	159	158	157	156	155	154	153	152	151	150	149	148	147	146
400	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131	130
416	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114
432	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98
448	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82
464	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66
480	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50
496	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34
512	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18
528	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
544	0															

1606

FIG 16(6)

1607

pki=6

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	18820	17368	15880	14246	12182	10125	8491	7000	5533	5389	5267	5137	4999	4863	4703	4541
16	4387	4226	4049	3818	3584	3344	3108	2929	2765	2610	2447	2284	2147	2012	1883	1758
32	1608	1571	1550	1522	1502	1478	1457	1430	1404	1374	1348	1328	1308	1254	1224	1179
48	1152	1125	1081	1051	1000	980	959	935	905	880	855	833	808	786	760	739
64	704	702	680	673	668	660	655	642	635	630	628	625	615	605	601	599
80	594	587	574	569	561	556	550	526	524	523	522	521	520	519	518	517
96	516	514	513	511	510	508	507	506	505	503	500	497	495	493	491	483
112	479	474	467	465	463	461	458	456	454	453	452	451	449	448	447	445
128	444	443	442	441	440	439	438	437	436	435	434	433	432	431	430	429
144	428	427	426	425	424	422	421	420	419	418	417	416	415	414	413	412
160	411	410	408	407	406	405	404	403	402	401	400	399	398	397	396	395
176	394	392	391	388	386	384	382	380	379	378	377	375	373	371	367	366
192	364	363	362	361	360	359	358	357	356	355	354	353	352	351	350	349
208	348	347	346	345	344	343	342	341	340	339	338	337	336	335	334	333
224	332	331	330	329	328	327	326	325	324	323	322	321	320	319	318	317
240	316	315	314	313	312	311	310	308	307	306	305	304	303	302	301	300
256	299	298	297	296	295	294	293	292	291	290	289	288	287	286	285	284
272	283	282	281	280	279	278	277	276	275	274	272	271	270	269	268	267
288	266	265	264	263	262	261	260	259	258	257	256	255	254	253	252	251
304	250	249	248	247	246	245	244	243	242	241	240	239	238	237	236	235
320	234	233	231	230	229	227	226	225	224	223	222	221	220	219	218	216
336	215	212	211	210	209	208	207	206	205	204	203	202	201	200	199	198
352	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181
368	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165
384	164	163	162	161	160	159	158	157	156	155	154	153	152	151	150	149
400	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133
416	132	131	130	129	128	127	126	125	124	123	122	121	120	119	118	117
432	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101
448	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85
464	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69
480	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53
496	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37
512	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21
528	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5
544	0															

FIG 16(7)

pki=7

1608

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	17955	16262	14706	13157	11504	9856	8311	6757	5082	4928	4798	4673	4541	4399	4212	4065
16	3938	3799	3627	3453	3270	3085	2908	2733	2592	2463	2307	2123	1983	1854	1728	1600
32	1440	1406	1386	1362	1343	1322	1302	1280	1256	1221	1197	1176	1155	1113	1092	1055
48	1033	1011	974	953	910	889	867	844	812	788	766	746	725	706	682	663
64	628	624	605	600	597	589	584	571	565	561	559	556	552	549	547	545
80	540	534	521	516	510	506	501	482	478	476	475	474	473	472	470	469
96	468	467	466	465	463	462	461	460	459	457	456	454	453	452	451	448
112	446	444	441	440	438	437	435	434	432	431	430	429	428	427	426	425
128	424	423	422	421	420	419	418	417	415	414	413	412	411	410	409	408
144	407	406	405	404	403	402	401	400	399	398	397	396	395	394	393	392
160	391	390	389	388	387	386	385	384	383	382	381	380	379	378	377	376
176	375	374	373	371	370	369	368	367	366	365	364	363	362	361	359	358
192	356	355	354	353	352	351	350	349	348	347	346	345	344	343	342	341
208	340	339	338	337	336	335	334	333	332	331	330	329	328	327	326	325
224	324	323	322	321	320	319	318	317	316	315	314	313	312	311	310	309
240	308	307	306	305	304	303	302	301	300	299	298	297	296	295	294	293
256	292	291	290	289	288	287	286	285	284	283	282	281	280	279	278	277
272	276	275	274	273	272	271	270	269	268	267	266	265	264	263	262	261
288	260	259	258	257	256	255	254	253	252	251	250	249	248	247	246	245
304	244	243	242	241	240	239	238	237	236	235	234	233	232	231	230	229
320	228	227	225	224	223	222	221	220	219	218	217	216	215	214	213	212
336	211	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195
352	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179
368	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163
384	162	161	160	159	158	157	156	155	154	153	152	151	150	149	148	147
400	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131
416	130	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115
432	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99
448	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83
464	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67
480	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51
496	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
512	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19
528	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
544	0															

FIG 16(8)

pki=8

1609

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	23611	21653	19758	17780	15542	13317	11370	9499	7545	7324	7126	6933	6732	6534	6295	6072
16	5871	5658	5412	5142	4840	4548	4277	4027	3815	3596	3374	3130	2927	2725	2531	2337
32	2123	2072	2038	1999	1969	1933	1897	1855	1819	1770	1730	1695	1663	1597	1558	1498
48	1467	1427	1366	1327	1260	1225	1191	1151	1098	1063	1024	985	949	917	881	849
64	795	792	765	755	748	737	728	704	692	683	680	677	670	661	657	653
80	645	635	615	606	595	585	577	551	546	544	543	542	541	539	538	537
96	535	534	532	530	528	526	525	523	522	520	517	513	511	509	507	500
112	496	493	486	484	482	479	476	474	472	470	468	467	466	465	464	462
128	460	459	457	455	453	452	451	450	448	447	446	445	444	443	442	441
144	440	439	438	437	436	435	434	433	432	431	430	428	427	426	425	424
160	423	422	420	419	418	417	415	414	413	411	410	409	408	407	406	405
176	404	402	400	397	396	394	392	391	390	389	388	387	384	382	379	377
192	374	373	372	371	370	369	368	367	366	365	364	362	361	360	359	357
208	356	354	353	352	351	350	349	348	347	346	345	344	342	341	340	339
224	338	337	336	335	334	333	332	331	330	329	328	327	326	325	324	323
240	322	321	320	319	318	316	315	313	312	311	310	309	308	307	306	305
256	304	303	302	301	300	299	298	297	296	295	294	293	292	291	290	289
272	288	287	286	284	283	282	281	279	278	277	275	274	272	271	270	269
288	268	267	266	265	264	263	262	261	260	259	258	257	256	255	254	253
304	252	251	250	249	248	247	245	244	243	242	241	240	239	238	237	236
320	235	234	232	231	230	229	228	227	226	225	224	223	222	221	220	217
336	216	213	212	211	210	209	208	207	206	205	204	203	202	200	199	198
352	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180
368	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164
384	163	162	161	160	159	158	157	156	155	154	153	152	151	150	149	148
400	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132
416	131	130	129	128	127	126	125	124	123	122	121	120	119	118	117	116
432	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100
448	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84
464	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68
480	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52
496	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36
512	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20
528	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
544	0															

FIG 16(9)

pki = 9

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	21499	20103	18641	16916	14421	11930	10235	8763	7392	7219	7069	6906	6740	6582	6400	6197
16	5987	5773	5564	5219	4881	4542	4202	3984	3777	3573	3368	3183	3025	2846	2684	2531
32	2361	2309	2277	2237	2206	2168	2134	2095	2054	2011	1976	1947	1916	1836	1789	1719
48	1673	1627	1561	1518	1437	1410	1381	1347	1303	1263	1226	1192	1155	1125	1085	1057
64	1009	1006	966	954	946	934	925	902	891	884	881	876	853	831	826	822
80	815	805	783	775	761	750	739	703	700	698	697	696	695	693	692	690
96	688	685	682	681	679	676	675	673	671	667	662	656	653	649	644	627
112	616	607	590	586	582	579	572	568	565	563	561	560	558	556	553	551
128	548	546	544	542	541	540	539	538	536	535	534	533	532	531	530	529
144	528	527	526	525	524	522	521	520	519	518	517	514	513	512	511	509
160	507	505	503	501	499	498	497	495	493	492	491	490	488	487	486	485
176	483	479	476	468	462	458	454	451	449	448	446	443	440	435	428	425
192	421	420	418	417	416	415	414	413	412	410	408	407	406	404	403	401
208	399	397	394	393	392	390	388	387	385	383	381	380	378	377	376	375
224	374	373	372	371	370	369	368	367	366	365	364	363	362	361	360	359
240	358	357	355	354	353	351	350	347	346	345	343	342	341	340	339	338
256	337	335	334	333	331	330	329	328	327	326	325	324	323	322	321	320
272	319	318	317	316	315	314	313	311	310	309	306	305	303	302	301	300
288	299	298	297	296	295	294	293	292	291	290	289	288	287	286	285	284
304	283	282	281	280	279	278	276	275	274	273	272	271	270	269	268	267
320	266	264	261	259	258	255	254	252	251	250	249	247	246	244	242	238
336	236	231	230	229	228	226	225	224	223	220	218	216	215	212	211	210
352	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191
368	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175
384	174	172	171	170	169	168	167	166	165	164	163	162	161	159	158	157
400	156	155	153	152	151	150	149	148	147	146	145	144	143	142	141	139
416	138	137	136	135	134	133	132	131	130	129	128	127	126	125	124	123
432	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107
448	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91
464	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75
480	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59
496	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43
512	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27
528	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
544	0															

1610

FIG 16(10)

pki=10

1611

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	26784	24861	22951	20916	18319	15791	13726	11818	9877	9625	9406	9166	8916	8671	8371	8119
16	7849	7562	7254	6890	6469	6059	5689	5384	5104	4834	4563	4254	4026	3773	3542	3302
32	3047	2979	2925	2870	2829	2778	2719	2660	2612	2538	2488	2433	2391	2288	2228	2130
48	2072	2012	1906	1853	1760	1714	1658	1602	1531	1481	1427	1366	1313	1272	1212	1164
64	1102	1097	1064	1050	1036	1018	1005	969	954	944	938	931	913	899	891	884
80	872	858	823	806	793	780	763	722	716	712	710	709	708	705	701	698
96	695	691	688	686	684	681	679	676	673	672	668	665	662	659	656	645
112	637	628	615	610	606	602	598	594	590	588	586	583	580	576	573	571
128	567	565	561	558	556	555	554	552	551	550	549	547	545	544	543	542
144	541	540	539	538	537	536	535	532	530	529	528	525	524	523	521	518
160	516	513	511	510	508	506	504	502	499	497	496	494	492	491	490	489
176	487	484	481	475	472	469	464	462	460	459	456	453	450	446	440	439
192	436	434	433	431	430	428	427	425	423	422	421	419	417	414	411	409
208	407	405	402	401	400	398	396	394	392	391	389	388	386	385	384	383
224	382	381	380	378	377	376	375	373	371	370	369	368	367	366	365	364
240	363	361	359	358	356	353	351	349	347	346	344	343	342	341	340	339
256	337	336	335	333	332	331	330	329	328	327	326	325	322	320	319	318
272	315	314	312	311	310	309	308	306	305	304	300	299	298	296	294	291
288	289	288	287	286	285	284	283	282	281	280	279	278	277	276	275	274
304	273	272	271	270	269	267	266	265	264	263	262	261	260	259	258	257
320	256	254	251	249	248	247	245	243	242	241	239	237	236	234	233	230
336	228	226	225	224	223	221	219	218	217	216	214	211	210	208	207	206
352	201	199	198	197	196	195	194	193	192	191	190	189	188	187	185	184
368	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168
384	167	165	164	163	162	161	160	159	158	157	156	155	154	153	152	151
400	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135
416	134	133	132	131	130	129	128	127	126	125	124	123	122	121	120	119
432	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103
448	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87
464	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71
480	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55
496	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39
512	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23
528	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
544	0															

FIG 16(11)

pki = 11

1612

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	20193	18370	16747	15140	13428	11703	10107	8481	6639	6436	6262	6103	5936	5764	5516	5322
16	5149	4972	4757	4546	4311	4066	3857	3637	3463	3293	3098	2846	2671	2502	2344	2176
32	1975	1934	1912	1881	1853	1823	1789	1757	1720	1668	1635	1604	1572	1513	1481	1430
48	1399	1365	1314	1286	1227	1191	1159	1122	1069	1036	1004	972	939	909	872	845
64	799	793	767	759	751	739	728	706	696	687	684	681	677	670	666	662
80	654	642	621	612	603	595	587	559	553	549	548	547	545	543	541	539
96	537	536	534	533	531	529	528	526	524	522	520	519	517	515	513	507
112	504	501	496	494	492	490	487	485	482	480	478	476	474	472	470	469
128	468	466	464	462	460	458	457	455	453	451	450	449	447	446	445	444
144	443	442	441	440	439	437	436	435	434	433	432	431	430	429	428	427
160	426	425	423	422	421	420	418	417	416	415	413	412	411	410	409	408
176	406	404	403	401	400	398	396	394	392	391	390	389	387	385	383	382
192	379	378	377	376	375	373	372	371	369	368	366	365	364	362	361	360
208	359	357	356	355	354	353	352	351	350	349	347	346	344	343	342	341
224	340	339	338	337	336	335	334	333	332	331	330	329	328	327	326	325
240	324	322	320	319	318	316	315	314	313	312	311	310	309	308	307	306
256	305	304	303	302	301	300	299	298	297	296	295	294	293	291	290	289
272	288	287	286	284	283	282	281	279	278	277	276	275	273	272	271	270
288	268	267	266	265	264	263	262	261	260	259	258	257	256	255	254	253
304	252	251	250	249	248	247	246	245	244	243	242	241	240	239	238	237
320	236	235	233	232	231	230	229	228	227	226	225	224	223	221	220	218
336	216	213	212	211	210	209	208	207	206	204	202	200	199	198	197	196
352	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179
368	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163
384	162	161	160	159	158	157	156	155	154	153	152	151	150	149	148	147
400	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131
416	130	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115
432	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99
448	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83
464	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67
480	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51
496	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
512	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19
528	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
544	0															

FIG 16(12)

1613

pki = 12

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	26367	24294	22296	20282	17965	15670	13696	11736	9641	9379	9134	8909	8664	8426	8110	7846
16	7594	7336	7005	6691	6311	5932	5602	5286	5032	4776	4516	4192	3947	3697	3445	3198
32	2936	2866	2823	2771	2721	2665	2609	2558	2501	2425	2374	2308	2265	2175	2123	2022
48	1974	1924	1832	1784	1697	1650	1597	1539	1462	1409	1350	1289	1235	1191	1136	1090
64	1028	1023	980	960	947	930	916	881	866	852	847	841	830	821	816	810
80	799	779	742	724	711	699	688	657	652	649	648	646	643	640	635	633
96	632	629	627	625	624	620	617	616	614	610	606	602	600	597	596	587
112	581	574	565	562	557	554	550	548	544	541	538	536	533	531	528	527
128	524	521	519	516	514	512	510	509	503	501	500	499	497	496	495	494
144	493	492	490	489	488	487	486	483	482	481	480	478	477	476	474	473
160	471	470	467	466	465	463	462	461	460	458	457	456	454	452	451	450
176	449	447	446	441	438	437	433	431	429	428	427	425	422	421	416	414
192	412	409	408	406	404	403	402	400	399	397	396	395	394	393	391	389
208	388	387	386	385	383	382	379	377	375	374	373	372	369	367	365	364
224	363	362	361	360	359	358	356	355	353	352	351	350	347	346	345	344
240	343	342	341	340	338	337	336	332	331	330	328	327	326	325	324	323
256	322	320	319	317	315	314	312	311	310	309	308	307	305	303	302	301
272	298	297	296	295	294	293	292	291	290	289	286	285	282	281	280	279
288	278	277	275	274	273	272	271	270	269	268	267	266	265	264	263	262
304	261	260	259	258	257	256	255	254	253	252	251	250	249	248	247	246
320	245	244	242	240	238	237	236	235	234	231	229	228	227	226	225	223
336	222	219	218	217	216	215	214	212	211	210	209	207	206	205	204	202
352	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184
368	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168
384	167	165	164	163	162	161	160	159	158	157	156	155	154	153	152	151
400	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135
416	134	133	132	131	130	129	128	127	126	125	124	123	122	121	120	119
432	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103
448	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87
464	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71
480	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55
496	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39
512	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23
528	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
544	0															

FIG 16(13)

1614

pki=13

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	25722	24316	22959	21323	18483	15636	14007	12628	11217	11004	10807	10594	10375	10191	9935	9689
16	9411	9160	8888	8407	7935	7467	6985	6715	6467	6190	5942	5700	5507	5279	5063	4873
32	4661	4590	4534	4474	4426	4368	4305	4241	4177	4103	4043	3987	3941	3808	3732	3609
48	3539	3470	3349	3272	3142	3097	3046	2987	2913	2855	2795	2734	2673	2628	2566	2517
64	2437	2428	2370	2347	2329	2307	2287	2236	2215	2196	2189	2178	2091	2007	1997	1991
80	1974	1952	1896	1877	1857	1840	1819	1754	1744	1739	1736	1733	1730	1726	1721	1717
96	1714	1704	1696	1692	1689	1686	1684	1680	1676	1669	1653	1636	1630	1621	1612	1548
112	1511	1475	1411	1402	1392	1387	1370	1355	1347	1343	1339	1336	1331	1327	1323	1315
128	1307	1304	1300	1295	1292	1289	1286	1283	1277	1274	1272	1270	1268	1266	1264	1263
144	1261	1259	1258	1256	1254	1251	1249	1247	1244	1242	1238	1233	1231	1228	1225	1218
160	1213	1207	1201	1198	1195	1192	1188	1184	1181	1178	1176	1174	1171	1169	1166	1163
176	1161	1152	1145	1121	1111	1101	1091	1085	1081	1076	1070	1061	1050	1037	1014	1008
192	998	995	993	990	987	984	983	981	977	974	970	965	962	958	954	946
208	940	936	930	927	924	921	916	912	909	907	905	903	899	896	894	892
224	890	887	885	883	882	879	877	875	873	872	871	869	868	867	865	863
240	861	859	855	853	851	845	843	833	830	828	823	821	818	816	812	809
256	805	801	798	796	793	791	789	787	786	784	783	781	778	775	772	769
272	765	761	758	755	753	751	748	743	741	737	727	725	720	718	716	713
288	710	709	708	707	706	704	702	701	699	698	697	696	695	694	692	690
304	689	688	687	681	678	674	669	667	666	665	664	663	662	661	660	659
320	658	653	642	638	634	624	619	614	611	601	596	592	589	585	580	569
336	565	546	543	541	537	529	521	516	511	500	494	486	482	468	465	461
352	451	448	441	438	435	433	429	426	424	422	419	416	414	412	409	403
368	401	397	395	393	391	389	388	386	385	383	382	380	379	377	375	373
384	371	365	364	363	362	360	358	357	356	353	352	350	349	345	344	342
400	339	337	329	327	325	324	322	319	318	317	314	312	311	309	307	299
416	298	294	293	292	290	289	287	285	283	281	280	279	278	276	275	273
432	269	268	267	266	265	264	263	261	260	258	256	253	252	249	248	246
448	245	242	240	239	238	237	236	235	233	231	229	227	225	223	221	218
464	214	213	212	211	210	209	208	207	206	204	203	201	200	196	194	191
480	188	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172
496	170	169	168	167	166	165	164	163	162	161	160	159	158	157	156	155
512	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137
528	136	135	134	133	132	131	130	129	128	127	126	125	124	123	122	121
544	0															

FIG 16(14)

1615

pki=14

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	35023	33000	31139	29182	26270	23436	21427	19496	17508	17136	16781	16423	16050	15697	15270	14891
16	14514	14112	13710	13135	12527	11948	11375	10929	10526	10146	9744	9324	9000	8567	8179	7836
32	7442	7316	7209	7081	6992	6878	6745	6613	6507	6369	6262	6165	6077	5876	5783	5578
48	5458	5345	5173	5064	4848	4743	4624	4492	4356	4237	4105	3977	3858	3754	3628	3515
64	3345	3326	3237	3189	3151	3101	3071	2986	2942	2903	2892	2872	2796	2709	2691	2676
80	2631	2584	2487	2452	2404	2362	2319	2195	2173	2162	2160	2154	2150	2143	2131	2123
96	2115	2103	2090	2082	2074	2062	2054	2048	2040	2028	2008	1992	1982	1972	1962	1903
112	1880	1850	1799	1789	1780	1770	1756	1736	1726	1718	1710	1704	1696	1690	1680	1668
128	1654	1647	1642	1633	1621	1617	1610	1603	1595	1591	1584	1580	1576	1574	1570	1566
144	1563	1559	1557	1552	1544	1539	1533	1529	1526	1520	1517	1507	1502	1496	1491	1484
160	1474	1467	1455	1451	1443	1440	1435	1430	1423	1415	1413	1409	1405	1402	1398	1394
176	1390	1381	1375	1355	1349	1339	1325	1314	1307	1302	1286	1278	1268	1254	1232	1227
192	1214	1208	1204	1199	1196	1194	1190	1188	1184	1175	1172	1162	1157	1145	1140	1130
208	1118	1113	1105	1101	1095	1092	1081	1077	1076	1074	1070	1065	1056	1050	1046	1041
224	1036	1032	1028	1025	1017	1012	1008	1005	999	995	994	991	990	989	985	983
240	978	972	968	966	960	948	946	931	929	926	916	914	911	908	899	895
256	887	884	879	876	872	867	863	859	857	855	853	847	843	836	831	828
272	822	820	816	809	807	802	798	794	791	788	774	772	763	759	757	750
288	747	743	739	737	736	731	729	725	722	721	720	719	718	716	715	713
304	712	711	710	702	696	691	682	680	678	677	675	672	670	668	667	665
320	664	655	639	631	625	617	613	605	601	595	589	581	575	567	559	549
336	543	519	515	511	507	499	489	484	482	471	465	455	451	438	434	429
352	413	409	405	401	399	397	393	391	390	388	385	381	377	374	370	363
368	362	361	358	357	355	354	352	349	348	347	345	344	342	341	339	335
384	334	324	323	322	321	316	315	312	311	310	309	305	303	299	297	296
400	289	287	279	275	274	272	268	264	262	261	259	257	255	251	249	247
416	246	242	241	239	235	234	233	232	231	229	228	226	225	224	222	221
432	218	217	216	215	214	213	212	211	209	208	206	203	202	200	199	197
448	194	192	191	189	188	187	186	185	182	180	179	177	176	172	170	169
464	164	163	161	160	159	157	156	154	152	151	150	148	147	145	143	141
480	137	134	133	132	131	130	129	128	127	125	123	122	121	120	119	118
496	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100
512	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	82
528	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66
544	0															

FIG 16(15)

1616

pki=15

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	34768	32700	30864	28851	25960	23061	21129	19296	17287	16902	16551	16168	15791	15444	15020	14599
16	14205	13822	13425	12855	12326	11756	11177	10781	10384	9975	9595	9176	8836	8492	8123	7808
32	7424	7267	7164	7025	6914	6805	6690	6567	6452	6307	6189	6080	5979	5774	5653	5475
48	5357	5236	5055	4943	4762	4662	4561	4463	4326	4181	4068	3961	3846	3749	3632	3528
64	3391	3378	3239	3202	3157	3112	3073	2972	2922	2886	2869	2851	2776	2694	2673	2663
80	2621	2566	2480	2451	2398	2360	2317	2224	2207	2194	2187	2182	2178	2169	2164	2155
96	2151	2140	2129	2121	2113	2104	2098	2087	2084	2075	2053	2030	2023	2008	1997	1948
112	1929	1894	1849	1836	1821	1810	1793	1778	1770	1764	1752	1747	1738	1727	1718	1704
128	1693	1690	1681	1672	1667	1665	1660	1653	1644	1641	1638	1633	1628	1624	1619	1610
144	1605	1601	1596	1593	1586	1577	1575	1572	1567	1561	1554	1539	1535	1528	1523	1518
160	1510	1499	1482	1478	1469	1464	1454	1446	1441	1436	1434	1427	1420	1416	1411	1405
176	1402	1388	1378	1355	1340	1331	1322	1310	1306	1299	1286	1275	1268	1252	1231	1226
192	1212	1207	1202	1198	1192	1190	1189	1184	1181	1175	1166	1161	1157	1152	1145	1136
208	1127	1121	1111	1106	1102	1099	1093	1086	1081	1079	1074	1071	1066	1060	1055	1050
224	1043	1041	1038	1035	1032	1028	1023	1018	1015	1009	1006	1004	999	996	994	989
240	986	980	974	972	969	957	954	945	938	934	930	927	924	921	914	910
256	902	895	890	883	879	877	872	867	864	862	860	857	854	849	846	843
272	834	829	823	818	817	812	810	801	799	790	777	770	763	761	759	757
288	752	749	748	742	739	734	731	730	728	727	725	724	723	722	720	719
304	717	714	713	710	708	707	699	697	695	693	691	689	688	687	686	685
320	684	675	660	653	646	635	627	622	617	604	599	589	583	575	572	556
336	547	523	516	509	503	496	489	484	480	471	468	461	454	442	435	429
352	410	408	399	396	392	389	384	381	378	375	371	368	365	364	359	353
368	350	348	345	344	343	342	341	340	339	337	335	333	332	330	329	328
384	325	316	315	314	313	312	309	308	306	301	299	298	297	292	289	288
400	285	284	279	277	274	271	268	265	264	261	259	258	257	254	253	250
416	246	243	241	240	239	238	236	235	232	230	229	228	227	224	222	221
432	217	215	214	213	212	211	210	209	208	207	205	202	201	199	197	195
448	194	192	190	188	185	184	183	182	181	178	175	172	171	170	169	167
464	164	163	162	161	160	159	158	157	156	155	154	153	152	147	145	142
480	139	137	136	135	134	133	132	131	129	128	127	126	125	124	123	121
496	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103
512	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87
528	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71
544	0															

FIG 16(16)

1617

pki=16

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	33834	31259	29286	27503	24983	22657	20714	18799	16591	16243	15873	15485	15110	14814	14291	13965
16	13558	13181	12680	12248	11754	11115	10664	10269	9926	9572	9231	8726	8385	8027	7676	7334
32	6901	6779	6704	6613	6526	6420	6306	6189	6091	5933	5819	5708	5611	5453	5344	5109
48	4997	4904	4716	4607	4470	4361	4226	4122	3963	3857	3736	3613	3516	3426	3333	3239
64	3123	3063	2993	2961	2933	2901	2860	2770	2738	2705	2683	2667	2609	2570	2554	2539
80	2502	2448	2339	2302	2272	2238	2206	2135	2104	2083	2068	2057	2048	2040	2008	2001
96	1997	1985	1972	1966	1953	1940	1933	1921	1911	1899	1876	1863	1851	1842	1835	1806
112	1787	1758	1724	1710	1698	1692	1679	1666	1660	1651	1646	1637	1626	1614	1608	1595
128	1586	1580	1572	1555	1544	1538	1530	1523	1501	1495	1493	1490	1485	1483	1478	1472
144	1469	1460	1455	1449	1447	1437	1435	1432	1430	1423	1415	1409	1407	1404	1401	1396
160	1391	1386	1379	1374	1369	1364	1359	1353	1350	1347	1341	1339	1334	1329	1324	1317
176	1315	1310	1304	1283	1278	1263	1254	1245	1240	1234	1224	1214	1204	1198	1184	1180
192	1171	1165	1159	1156	1153	1142	1135	1131	1124	1120	1114	1106	1099	1094	1089	1080
208	1072	1067	1059	1052	1048	1042	1036	1027	1022	1017	1011	1006	1001	990	985	980
224	973	968	966	964	961	952	949	946	941	938	935	933	931	929	925	922
240	918	914	910	909	906	901	898	887	883	880	879	878	875	873	866	863
256	860	854	849	846	844	840	836	832	828	824	822	817	814	810	806	804
272	801	789	782	775	772	765	761	758	755	752	746	741	733	728	724	721
288	718	715	713	711	709	704	701	699	697	694	693	692	691	690	688	686
304	685	684	681	680	676	673	667	665	663	662	660	658	656	655	654	653
320	651	646	640	634	630	625	619	611	606	598	593	585	582	573	562	548
336	540	526	521	516	513	505	496	491	484	475	469	459	456	445	433	427
352	415	410	407	402	400	397	393	391	388	384	383	380	377	376	373	368
368	366	363	362	360	358	355	353	350	348	345	343	339	338	337	336	334
384	333	325	324	322	320	318	316	314	311	308	307	305	304	301	300	298
400	295	294	289	286	283	282	281	278	276	275	274	272	270	268	267	262
416	257	255	253	250	248	247	245	243	241	240	238	236	235	233	231	230
432	228	227	226	225	224	223	222	221	220	218	217	215	214	212	210	207
448	205	204	201	199	197	196	195	192	188	185	184	180	179	177	175	172
464	169	168	167	166	165	164	163	162	161	160	158	156	155	152	151	149
480	148	147	146	145	143	142	141	140	139	137	136	135	134	133	132	130
496	128	127	126	125	124	123	122	121	120	119	118	116	115	114	113	112
512	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
528	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
544	0															

FIG 16(17)

1618

pki = 17

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	36131	33941	31694	29678	26923	24502	22233	20170	18256	17912	17594	17201	16832	16494	15928	15475
16	14984	14545	13989	13516	12933	12148	11626	11145	10762	10370	10023	9557	9186	8823	8493	8149
32	7731	7615	7528	7442	7349	7264	7149	7026	6935	6781	6661	6537	6452	6284	6147	5868
48	5704	5600	5405	5280	5122	4992	4872	4745	4582	4475	4337	4205	4110	3979	3878	3770
64	3655	3591	3531	3490	3449	3416	3387	3287	3256	3227	3202	3182	3108	3056	3020	2994
80	2948	2890	2793	2747	2703	2661	2624	2550	2537	2526	2512	2505	2500	2495	2467	2459
96	2452	2435	2425	2418	2401	2386	2379	2368	2358	2347	2319	2298	2281	2269	2257	2215
112	2191	2152	2106	2084	2062	2055	2034	2016	2003	1994	1987	1979	1969	1958	1951	1940
128	1929	1920	1902	1893	1888	1883	1879	1868	1857	1852	1847	1841	1838	1835	1830	1829
144	1826	1819	1813	1806	1801	1796	1793	1785	1778	1773	1769	1764	1758	1755	1751	1741
160	1734	1729	1717	1706	1703	1700	1691	1684	1681	1674	1671	1668	1661	1654	1643	1633
176	1624	1611	1605	1586	1576	1557	1536	1515	1502	1495	1488	1475	1460	1443	1425	1412
192	1401	1396	1389	1384	1377	1367	1364	1361	1354	1343	1334	1323	1316	1313	1306	1295
208	1287	1274	1263	1256	1239	1233	1226	1220	1215	1200	1197	1194	1187	1184	1180	1175
224	1174	1171	1167	1164	1163	1155	1151	1145	1141	1137	1134	1133	1130	1127	1124	1122
240	1119	1115	1112	1107	1105	1098	1095	1084	1079	1075	1068	1062	1057	1054	1048	1045
256	1038	1033	1030	1027	1018	1015	1008	1003	1001	998	995	988	984	978	972	965
272	956	944	939	926	923	918	910	903	900	894	886	881	873	868	864	857
288	850	848	845	842	839	838	833	832	830	828	827	826	825	824	822	820
304	817	815	812	809	806	795	786	784	783	781	780	779	778	777	775	774
320	769	761	747	734	729	722	717	706	703	690	680	675	673	663	656	642
336	634	623	620	614	607	599	585	578	572	559	555	543	536	518	509	502
352	488	483	478	475	474	471	468	461	460	455	450	444	439	437	433	426
368	422	420	417	415	411	408	400	397	396	394	391	388	387	383	380	375
384	372	365	363	362	361	359	358	355	354	348	346	343	342	335	333	330
400	329	326	321	316	315	313	311	310	309	306	303	299	298	295	290	286
416	281	278	277	274	272	271	269	266	263	261	258	255	254	253	250	249
432	246	244	242	241	239	238	237	235	234	232	230	229	228	225	224	223
448	221	220	219	216	215	213	212	210	209	207	205	203	202	200	199	195
464	186	185	184	183	182	180	179	177	175	174	173	172	171	169	168	165
480	162	161	160	159	158	157	156	155	154	153	152	150	149	148	147	146
496	143	142	141	140	139	138	137	136	135	133	132	131	130	129	128	126
512	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109
528	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93
544	0															

FIG 16(18)

1619

pki=18

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	37328	35603	33801	31650	27640	25095	22428	20566	18919	18646	18347	18009	17711	17404	17024	16583
16	16046	15610	15083	14478	13898	12708	11998	11521	11114	10750	10405	10033	9742	9376	9073	8744
32	8465	8355	8254	8169	8096	7989	7890	7759	7658	7530	7422	7323	7241	7035	6869	6593
48	6401	6290	6100	5955	5784	5700	5587	5474	5311	5211	5120	4969	4854	4755	4661	4568
64	4471	4454	4383	4348	4309	4259	4229	4143	4106	4077	4068	4045	3737	3669	3617	3604
80	3567	3515	3412	3360	3316	3289	3253	3180	3166	3163	3156	3151	3147	3142	3131	3125
96	3116	3100	3094	3086	3077	3066	3063	3054	3045	3027	2964	2939	2921	2904	2891	2778
112	2751	2560	2489	2453	2410	2398	2371	2356	2338	2326	2322	2315	2311	2302	2291	2270
128	2257	2246	2238	2233	2230	2224	2219	2216	2209	2206	2203	2200	2198	2195	2193	2190
144	2187	2184	2180	2179	2176	2170	2164	2155	2151	2147	2138	2133	2125	2122	2120	2108
160	2102	2092	2084	2078	2071	2068	2060	2052	2047	2041	2038	2033	2029	2020	2012	2006
176	2001	1988	1977	1926	1901	1856	1833	1808	1793	1788	1778	1767	1755	1738	1715	1695
192	1681	1676	1668	1662	1658	1651	1646	1640	1634	1625	1612	1605	1600	1597	1590	1579
208	1572	1555	1538	1526	1519	1514	1505	1500	1492	1479	1475	1472	1467	1464	1461	1459
224	1454	1452	1448	1445	1443	1441	1439	1434	1431	1428	1427	1425	1422	1420	1415	1413
240	1411	1407	1401	1396	1393	1390	1384	1367	1364	1358	1351	1345	1342	1339	1333	1331
256	1326	1318	1315	1313	1308	1306	1303	1300	1297	1291	1288	1283	1280	1274	1272	1267
272	1264	1255	1248	1243	1237	1235	1230	1225	1220	1212	1205	1198	1190	1185	1181	1178
288	1173	1171	1168	1166	1165	1163	1160	1158	1156	1155	1154	1152	1151	1149	1146	1141
304	1139	1136	1135	1130	1122	1104	1084	1082	1081	1080	1077	1076	1073	1071	1070	1068
320	1065	1056	1039	1027	1022	1012	1007	997	994	947	938	928	920	913	908	894
336	890	864	856	850	839	832	824	818	813	773	759	729	723	687	680	672
352	663	660	651	646	641	638	635	627	624	616	610	593	588	584	581	558
368	555	552	550	549	546	545	542	540	537	535	533	529	528	522	520	517
384	512	503	501	498	497	496	494	493	491	482	480	479	478	469	468	467
400	465	462	455	450	448	447	445	440	437	434	426	418	415	413	411	402
416	400	393	392	391	388	385	382	381	378	376	372	370	368	366	364	363
432	355	353	352	351	350	349	348	347	346	341	338	325	324	318	316	315
448	314	311	310	309	308	307	305	304	303	298	293	285	284	282	279	268
464	264	262	261	259	257	256	255	253	252	247	244	241	240	235	234	228
480	225	224	223	221	219	218	217	215	214	210	209	207	206	205	204	202
496	200	199	198	197	196	195	194	192	191	190	189	188	187	186	185	184
512	182	181	180	179	178	177	176	175	173	172	171	170	169	168	167	166
528	164	163	162	161	160	159	158	157	156	155	154	153	152	151	150	149
544	0															

FIG 16(19)

pki = 19

1620

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	36571	35205	33734	31970	26412	23014	20626	19199	17863	17616	17378	17032	16770	16532	16224	15917
16	15383	15041	14637	13939	13319	12105	11231	10887	10502	10155	9831	9544	9306	8970	8717	8434
32	8185	8104	8016	7944	7875	7779	7692	7565	7478	7370	7284	7198	7152	6954	6837	6575
48	6436	6343	6176	6020	5849	5797	5695	5617	5503	5417	5335	5197	5090	5030	4951	4887
64	4788	4778	4709	4679	4640	4608	4580	4515	4475	4451	4443	4425	3946	3766	3734	3724
80	3691	3664	3568	3539	3506	3480	3446	3372	3362	3356	3351	3348	3345	3341	3336	3329
96	3322	3294	3286	3279	3274	3267	3262	3257	3252	3242	3159	3127	3114	3099	3079	2923
112	2873	2682	2525	2501	2469	2458	2422	2396	2380	2375	2370	2363	2358	2354	2347	2313
128	2299	2294	2288	2281	2277	2272	2268	2265	2260	2257	2254	2253	2251	2248	2247	2243
144	2241	2240	2238	2235	2234	2226	2223	2217	2214	2210	2200	2192	2187	2182	2177	2164
160	2156	2142	2135	2129	2121	2118	2109	2106	2104	2099	2097	2093	2091	2088	2083	2080
176	2076	2060	2049	1989	1968	1912	1893	1876	1864	1860	1852	1827	1808	1780	1744	1725
192	1708	1705	1700	1695	1693	1689	1687	1681	1674	1670	1661	1654	1649	1644	1640	1623
208	1614	1594	1583	1574	1567	1563	1549	1544	1537	1534	1528	1524	1521	1519	1515	1510
224	1506	1504	1501	1497	1495	1491	1488	1486	1483	1481	1480	1478	1476	1475	1474	1472
240	1470	1465	1461	1458	1453	1446	1441	1423	1420	1412	1405	1397	1393	1392	1387	1382
256	1377	1372	1366	1363	1357	1353	1350	1345	1344	1342	1339	1336	1332	1329	1325	1321
272	1316	1302	1295	1286	1283	1279	1275	1268	1262	1251	1235	1231	1219	1217	1215	1211
288	1207	1205	1203	1201	1198	1196	1195	1193	1192	1190	1189	1188	1187	1186	1185	1181
304	1179	1178	1177	1166	1160	1147	1126	1124	1122	1121	1118	1117	1116	1115	1114	1113
320	1112	1101	1081	1069	1060	1045	1038	1026	1023	975	964	948	937	931	926	913
336	909	874	870	864	857	849	838	832	826	792	780	752	743	705	700	695
352	679	672	662	656	653	648	645	640	638	634	624	610	602	599	592	577
368	573	570	567	564	561	559	558	554	551	549	546	543	540	533	531	529
384	527	516	515	514	512	509	508	507	506	497	496	495	494	485	483	481
400	476	473	463	459	458	457	454	451	449	446	438	430	427	422	418	408
416	407	399	397	396	394	393	392	390	388	385	382	381	379	376	375	372
432	366	365	364	363	361	360	359	358	357	353	350	338	337	331	330	329
448	328	324	321	320	319	317	316	314	312	308	305	300	296	293	290	280
464	276	274	273	272	271	270	268	267	265	260	258	255	253	248	244	236
480	233	231	230	229	228	227	226	225	224	222	219	217	216	213	212	209
496	206	205	204	203	202	201	200	199	198	197	196	194	193	192	191	189
512	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171
528	169	168	167	166	165	164	163	162	161	160	159	158	157	156	155	154
544	0															

FIG 16(20)

1621

pki=20

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	36539	35236	33760	31424	28150	22640	20855	19383	17989	17771	17488	17204	16841	16604	16326	15984
16	15625	15271	14906	13999	12788	12188	11518	11114	10748	10227	9927	9613	9354	9077	8741	8494
32	8274	8171	8107	8026	7968	7860	7732	7642	7547	7425	7344	7248	7184	7012	6870	6707
48	6604	6470	6196	6082	5882	5820	5730	5640	5532	5442	5327	5236	5151	5085	5002	4912
64	4825	4814	4740	4698	4678	4644	4611	4515	4475	4444	4435	4401	4233	3750	3729	3721
80	3697	3662	3592	3568	3532	3498	3465	3388	3380	3375	3371	3369	3367	3364	3357	3354
96	3343	3331	3299	3293	3287	3283	3279	3273	3265	3248	3221	3184	3172	3142	3112	2958
112	2779	2720	2566	2544	2532	2524	2491	2409	2393	2386	2379	2373	2367	2360	2350	2334
128	2300	2295	2291	2285	2283	2279	2276	2275	2271	2268	2266	2264	2263	2261	2259	2257
144	2254	2252	2249	2246	2243	2238	2234	2229	2225	2219	2215	2204	2199	2190	2183	2165
160	2153	2145	2125	2120	2116	2110	2103	2091	2086	2078	2075	2072	2070	2067	2064	2061
176	2057	2044	2026	1985	1956	1946	1926	1916	1911	1900	1883	1863	1816	1795	1724	1713
192	1696	1690	1686	1683	1680	1678	1676	1672	1670	1666	1661	1652	1649	1644	1636	1627
208	1610	1601	1584	1580	1578	1570	1561	1550	1546	1542	1538	1534	1531	1527	1523	1519
224	1517	1514	1511	1509	1508	1506	1504	1502	1499	1497	1496	1495	1493	1491	1489	1487
240	1486	1482	1480	1477	1476	1466	1462	1450	1438	1435	1427	1425	1423	1419	1415	1411
256	1401	1397	1391	1385	1381	1379	1374	1371	1369	1367	1366	1362	1360	1356	1352	1345
272	1337	1330	1326	1320	1316	1314	1308	1302	1295	1289	1268	1264	1258	1256	1254	1251
288	1248	1246	1245	1243	1241	1240	1237	1235	1231	1230	1229	1228	1227	1226	1225	1223
304	1222	1220	1219	1200	1187	1180	1169	1168	1166	1164	1161	1159	1158	1157	1156	1155
320	1154	1149	1127	1120	1114	1062	1051	1040	1034	1009	1002	995	992	984	976	958
336	952	910	906	902	898	860	830	815	801	791	784	773	769	739	735	731
352	719	714	692	686	682	677	660	652	645	642	634	628	621	618	614	600
368	596	591	589	588	586	584	581	578	575	573	571	568	566	564	560	559
384	552	541	539	538	537	530	528	527	523	521	519	517	515	505	503	501
400	497	495	483	480	478	475	469	464	462	461	456	452	451	449	447	434
416	430	426	424	422	420	418	416	413	411	408	407	405	404	402	400	399
432	392	391	390	389	388	387	386	384	383	381	379	377	375	371	370	369
448	367	360	358	357	356	354	351	350	347	344	341	339	338	335	333	330
464	326	325	324	323	322	321	320	316	314	313	311	310	309	303	299	293
480	288	286	285	284	283	282	281	279	277	276	275	273	272	271	270	268
496	266	265	264	263	262	261	260	259	257	256	255	254	253	252	251	250
512	247	246	245	244	243	242	241	240	239	238	237	236	235	234	233	232
528	231	230	229	228	227	226	225	224	223	222	221	220	219	218	217	216
544	0															

FIG 16(21)

1622

pki=21

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	37343	35792	33995	31291	28680	24772	22639	20806	19119	18828	18482	18166	17784	17461	17079	16720
16	16361	15956	15488	14707	13533	12927	12320	11788	11329	10812	10395	9994	9709	9403	9084	8775
32	8493	8385	8311	8217	8131	8012	7864	7747	7655	7499	7388	7276	7188	7018	6895	6696
48	6599	6410	6113	5942	5722	5653	5553	5449	5311	5214	5099	4998	4915	4826	4737	4658
64	4544	4530	4461	4422	4385	4340	4291	4177	4126	4089	4075	4026	3975	3677	3653	3645
80	3615	3579	3499	3465	3427	3390	3354	3288	3274	3265	3260	3257	3252	3246	3238	3228
96	3213	3199	3170	3161	3152	3146	3140	3133	3119	3092	3075	3046	3028	2982	2945	2875
112	2704	2675	2583	2569	2555	2536	2504	2442	2422	2414	2403	2395	2386	2375	2369	2357
128	2334	2326	2316	2311	2306	2301	2293	2288	2281	2278	2275	2272	2270	2267	2265	2263
144	2260	2258	2255	2252	2247	2242	2236	2232	2221	2212	2207	2199	2193	2180	2168	2156
160	2135	2124	2111	2104	2099	2091	2082	2067	2057	2048	2042	2037	2033	2029	2027	2021
176	2018	2003	1989	1965	1942	1936	1918	1907	1901	1890	1872	1842	1796	1776	1724	1710
192	1693	1687	1681	1675	1671	1668	1665	1659	1652	1647	1640	1636	1630	1626	1619	1610
208	1596	1586	1571	1562	1556	1548	1543	1535	1528	1525	1519	1514	1509	1507	1505	1502
224	1497	1494	1490	1488	1486	1484	1481	1479	1478	1477	1476	1473	1471	1469	1466	1463
240	1462	1456	1451	1447	1441	1429	1423	1411	1403	1401	1397	1392	1389	1381	1370	1362
256	1353	1346	1341	1337	1334	1330	1326	1321	1318	1316	1314	1308	1305	1303	1299	1294
272	1286	1279	1274	1268	1264	1258	1252	1246	1236	1229	1221	1216	1209	1206	1204	1198
288	1195	1192	1189	1187	1185	1184	1182	1180	1179	1177	1176	1175	1174	1172	1171	1169
304	1168	1165	1163	1146	1129	1127	1122	1120	1119	1117	1113	1111	1109	1107	1106	1105
320	1103	1097	1083	1079	1074	1024	1012	1002	998	992	975	970	962	953	940	923
336	917	879	871	868	865	830	799	791	780	769	757	745	736	718	710	705
352	694	688	667	661	658	653	642	631	623	622	619	615	613	610	602	597
368	593	587	581	578	575	573	568	564	559	558	555	553	551	549	545	543
384	538	531	530	529	528	519	516	515	511	509	504	503	501	495	494	492
400	491	490	480	479	475	474	468	462	460	458	454	452	451	449	444	441
416	437	431	430	428	425	423	421	416	411	410	405	403	399	396	393	391
432	384	383	382	381	380	378	376	375	374	373	370	369	367	365	364	362
448	361	356	353	352	351	348	347	346	344	342	340	338	337	334	332	327
464	324	322	321	320	319	316	315	312	311	309	307	306	305	299	296	295
480	292	289	288	287	286	285	283	281	279	278	277	276	275	272	271	268
496	265	264	263	262	261	260	259	257	256	255	254	253	252	250	249	248
512	245	244	243	242	241	240	239	238	237	236	235	234	233	232	231	230
528	228	227	226	225	224	223	222	221	220	219	218	217	216	215	214	213
544	0															

FIG 16(22)

1623

pki=22

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	36094	34208	32055	29807	27391	24560	22585	20417	18295	17849	17524	17193	16808	16443	15961	15602
16	15186	14790	14306	13787	13051	12496	12010	11416	10966	10522	10063	9485	9170	8803	8456	8124
32	7781	7654	7552	7448	7313	7209	7083	6947	6829	6675	6560	6418	6277	6108	6002	5801
48	5686	5515	5253	5124	4951	4860	4738	4622	4464	4380	4266	4157	4054	3968	3865	3771
64	3653	3640	3565	3532	3488	3454	3408	3316	3258	3217	3198	3169	3124	3043	3028	3009
80	2977	2946	2858	2815	2784	2750	2711	2640	2598	2585	2578	2573	2563	2552	2547	2531
96	2519	2507	2495	2484	2473	2467	2459	2450	2436	2425	2414	2399	2386	2364	2339	2289
112	2247	2220	2170	2161	2152	2131	2114	2084	2069	2059	2048	2039	2023	2009	1998	1985
128	1971	1963	1951	1920	1913	1907	1900	1885	1872	1866	1863	1859	1853	1852	1849	1847
144	1838	1831	1828	1821	1811	1806	1798	1793	1781	1777	1773	1764	1757	1748	1739	1728
160	1717	1710	1697	1692	1688	1681	1673	1663	1656	1647	1643	1638	1629	1624	1617	1612
176	1607	1595	1584	1563	1554	1543	1528	1522	1513	1498	1481	1464	1441	1428	1406	1394
192	1380	1373	1368	1360	1354	1344	1339	1335	1326	1319	1312	1306	1301	1294	1285	1273
208	1267	1258	1249	1246	1243	1238	1233	1228	1224	1217	1210	1207	1202	1196	1189	1183
224	1174	1172	1167	1163	1156	1152	1146	1143	1138	1134	1131	1127	1124	1122	1121	1119
240	1117	1111	1106	1102	1099	1092	1088	1076	1073	1069	1060	1057	1052	1044	1037	1032
256	1023	1018	1012	1005	1002	997	994	989	986	983	980	976	971	966	961	956
272	952	949	944	940	939	932	927	922	918	914	905	900	892	888	883	877
288	873	872	869	865	864	862	859	858	857	854	853	852	851	850	848	847
304	845	842	840	827	820	817	813	811	810	808	806	803	802	800	799	798
320	796	791	777	770	765	751	740	730	723	714	705	700	697	688	675	666
336	660	637	628	622	615	601	587	579	572	557	549	534	525	514	509	502
352	492	488	477	473	472	470	462	457	453	451	444	440	436	432	423	421
368	416	411	405	404	400	399	398	393	391	387	383	380	379	378	375	370
384	366	360	359	358	355	352	347	346	344	341	339	335	332	326	325	324
400	323	318	315	312	309	308	305	304	302	298	294	292	287	285	284	279
416	275	274	272	271	269	268	267	264	262	261	259	258	256	255	252	249
432	242	241	240	239	238	237	235	234	233	231	229	228	227	225	223	222
448	221	217	214	213	212	211	209	208	207	204	201	199	197	195	192	189
464	184	183	181	180	178	177	175	174	173	172	169	168	167	164	159	156
480	152	148	147	146	145	144	143	142	141	140	137	135	134	132	131	130
496	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114
512	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97
528	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81
544	0															

FIG 16(23)

pki=23

1624

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	33706	31512	29542	27668	25307	22825	21024	19142	16586	16142	15793	15427	15071	14725	14191	13873
16	13492	13115	12717	12261	11658	11121	10677	10234	9877	9462	9144	8606	8304	7937	7597	7221
32	6877	6762	6678	6582	6488	6402	6280	6181	6075	5914	5800	5683	5572	5426	5325	5123
48	5030	4918	4700	4594	4431	4345	4238	4132	3949	3855	3760	3662	3557	3478	3387	3306
64	3184	3161	3095	3061	3027	2994	2949	2841	2791	2754	2737	2722	2679	2632	2618	2598
80	2573	2539	2431	2389	2352	2315	2276	2204	2147	2123	2116	2107	2099	2089	2079	2071
96	2066	2052	2043	2032	2021	2013	2007	2001	1996	1987	1973	1957	1948	1939	1924	1893
112	1859	1836	1802	1794	1784	1770	1751	1731	1720	1710	1704	1695	1683	1674	1669	1660
128	1649	1645	1639	1618	1607	1599	1593	1582	1557	1554	1549	1547	1538	1534	1532	1529
144	1526	1522	1517	1512	1507	1501	1498	1493	1490	1482	1477	1471	1467	1462	1456	1448
160	1440	1430	1422	1415	1409	1401	1393	1385	1379	1371	1366	1359	1354	1351	1346	1340
176	1332	1321	1316	1305	1297	1283	1278	1267	1259	1254	1246	1237	1224	1216	1193	1187
192	1175	1171	1166	1161	1160	1156	1154	1151	1148	1145	1139	1134	1127	1124	1122	1113
208	1105	1099	1093	1088	1084	1079	1076	1072	1068	1065	1060	1058	1054	1046	1043	1040
224	1034	1031	1028	1024	1019	1014	1010	1006	1002	999	998	994	991	988	985	982
240	979	976	972	969	964	958	952	946	941	936	931	928	926	920	912	909
256	900	892	887	882	879	874	869	866	863	861	859	848	840	837	835	831
272	824	816	811	804	803	800	795	790	785	782	770	767	764	763	760	755
288	754	753	750	747	745	743	741	736	732	731	730	729	728	727	724	723
304	722	720	718	712	705	700	697	694	692	691	689	686	685	684	683	682
320	680	672	656	647	642	631	619	613	610	603	596	588	585	576	568	555
336	547	532	527	520	514	508	493	487	480	472	463	451	445	435	430	427
352	420	414	410	408	406	405	400	396	394	392	389	383	381	377	374	370
368	366	364	362	359	355	352	351	350	349	348	343	339	337	334	332	329
384	327	322	321	319	318	316	313	311	309	306	303	302	300	296	294	293
400	285	282	275	272	270	269	267	265	264	263	261	258	256	255	253	250
416	249	246	244	242	241	240	239	237	236	234	232	231	230	229	228	226
432	220	218	217	215	213	212	211	210	209	208	207	206	205	203	201	200
448	199	198	194	193	190	188	185	184	182	181	180	178	177	175	173	170
464	168	166	164	163	162	160	158	157	156	154	153	152	151	149	145	142
480	140	139	138	137	136	135	134	131	130	128	127	126	125	123	122	121
496	118	117	116	115	114	113	112	111	110	109	107	106	105	104	102	101
512	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85
528	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68
544	0															

FIG 16(24)

1625

pki = 24

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	29583	27193	25005	22830	20502	18176	15997	13813	11431	11092	10827	10558	10284	9980	9608	9295
16	9028	8724	8380	8030	7654	7276	6926	6581	6273	6005	5693	5325	5022	4745	4479	4214
32	3876	3783	3724	3651	3585	3525	3465	3394	3321	3235	3164	3105	3038	2940	2870	2779
48	2710	2640	2547	2477	2379	2314	2255	2185	2098	2025	1953	1893	1832	1767	1693	1635
64	1543	1529	1474	1455	1437	1411	1393	1356	1331	1313	1304	1294	1275	1256	1246	1237
80	1218	1193	1155	1137	1111	1094	1075	1021	1007	1000	997	994	991	986	979	974
96	971	967	963	959	954	949	946	942	937	931	926	919	914	908	902	891
112	881	871	860	854	848	843	837	832	827	822	818	815	810	805	801	797
128	793	790	785	778	774	771	768	765	758	755	753	751	748	746	744	742
144	740	737	735	733	730	726	724	721	718	715	713	710	707	705	703	700
160	697	694	690	687	684	682	679	677	674	671	668	666	663	660	657	655
176	652	647	643	638	634	630	626	622	619	616	612	608	604	600	595	592
192	587	584	582	579	577	574	572	569	566	563	561	558	556	553	550	546
208	543	540	537	535	533	530	527	525	522	519	516	514	510	507	505	503
224	501	499	497	495	494	491	489	487	484	482	481	480	478	477	475	473
240	472	469	467	466	464	461	459	456	454	453	450	448	446	444	442	440
256	438	436	434	432	430	428	426	424	423	422	420	418	416	414	412	411
272	409	407	405	403	401	399	397	394	392	390	386	384	381	379	378	376
288	373	372	370	369	368	366	365	364	362	361	360	359	358	357	356	355
304	354	353	352	350	348	346	344	343	342	341	340	339	338	337	336	335
320	334	330	325	322	319	316	313	310	308	304	301	298	296	292	289	284
336	281	274	272	270	267	264	261	258	255	251	248	245	243	239	237	234
352	228	227	225	224	223	222	221	220	219	218	217	216	215	214	212	210
368	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194
384	193	190	189	188	187	186	185	184	183	182	181	180	179	177	176	175
400	172	171	169	168	167	166	165	164	163	162	161	160	159	158	157	155
416	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139
432	138	137	136	135	134	133	132	131	130	129	128	127	126	125	124	123
448	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107
464	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91
480	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75
496	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59
512	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43
528	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27
544	0															

FIG 16(25)

pki=25

1626

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	39906	37160	34924	32771	30431	28096	25948	23695	20957	20373	19958	19508	19050	18545	17996	17506
16	17108	16631	16164	15646	15148	14654	14138	13669	13189	12793	12306	11759	11258	10802	10352	9936
32	9354	9125	8991	8809	8671	8540	8413	8253	8070	7893	7731	7607	7466	7255	7120	6954
48	6817	6682	6512	6374	6163	6023	5898	5739	5559	5376	5219	5094	4963	4828	4642	4507
64	4276	4249	4085	4031	3982	3902	3852	3753	3675	3622	3606	3589	3560	3533	3516	3500
80	3447	3373	3276	3228	3147	3097	3043	2880	2853	2834	2825	2816	2806	2793	2777	2765
96	2757	2746	2736	2724	2713	2701	2693	2682	2671	2659	2647	2631	2620	2608	2596	2574
112	2557	2540	2518	2507	2495	2484	2469	2458	2445	2434	2423	2416	2404	2393	2382	2371
128	2359	2351	2339	2323	2310	2301	2291	2282	2263	2254	2247	2238	2228	2223	2217	2209
144	2201	2193	2186	2180	2171	2157	2151	2142	2135	2128	2122	2113	2106	2100	2094	2085
160	2078	2069	2056	2048	2038	2032	2023	2017	2008	1999	1992	1987	1980	1973	1964	1958
176	1950	1936	1929	1916	1907	1898	1888	1878	1871	1863	1853	1842	1832	1823	1811	1804
192	1790	1783	1777	1768	1761	1754	1749	1741	1731	1722	1716	1706	1700	1690	1682	1670
208	1661	1654	1646	1640	1634	1626	1617	1611	1604	1597	1588	1582	1568	1560	1553	1546
224	1538	1531	1523	1518	1513	1503	1493	1485	1476	1469	1464	1459	1451	1447	1442	1436
240	1432	1422	1414	1410	1404	1394	1390	1379	1374	1370	1359	1353	1347	1343	1335	1329
256	1321	1313	1307	1301	1292	1286	1279	1273	1269	1265	1260	1253	1247	1239	1233	1228
272	1220	1213	1207	1200	1196	1190	1184	1174	1169	1164	1153	1149	1138	1133	1129	1120
288	1110	1106	1099	1093	1090	1082	1077	1073	1066	1062	1061	1059	1058	1056	1053	1050
304	1049	1046	1044	1040	1037	1034	1030	1027	1024	1023	1020	1017	1015	1013	1011	1010
320	1006	992	977	967	955	945	937	926	917	906	896	887	878	864	852	836
336	822	798	788	780	768	759	747	736	726	713	701	688	680	665	657	647
352	624	621	617	613	610	607	603	600	597	594	590	586	583	580	576	572
368	569	566	562	559	555	553	550	547	544	541	538	535	532	529	526	522
384	518	507	504	502	498	495	491	488	485	481	478	474	472	467	464	461
400	451	448	443	439	436	434	431	428	426	423	420	417	415	412	409	405
416	402	398	395	393	390	388	385	382	379	376	373	370	367	364	362	359
432	355	352	350	348	346	345	343	341	339	337	335	333	332	330	328	325
448	321	317	314	312	309	307	304	301	298	295	292	288	285	282	279	276
464	271	269	267	265	263	262	260	258	256	254	252	250	248	245	242	239
480	235	232	231	230	228	227	225	223	221	219	217	215	213	211	209	207
496	204	203	202	201	200	199	198	197	196	195	194	193	192	190	189	187
512	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170
528	168	167	166	165	164	163	162	161	160	159	158	157	156	155	154	153
544	0															

FIG 16(26)

1627

pki=26

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	41828	39150	36762	34403	31788	29163	26802	24416	21740	21195	20727	20254	19768	19263	18637	18146
16	17679	17177	16606	16055	15426	14802	14254	13692	13194	12723	12230	11600	11101	10612	10136	9661
32	9119	8930	8789	8625	8478	8335	8186	8021	7861	7657	7490	7342	7191	7001	6849	6642
48	6488	6338	6132	5981	5792	5643	5494	5330	5127	4966	4797	4648	4509	4356	4196	4055
64	3869	3833	3717	3661	3608	3540	3485	3380	3311	3253	3231	3208	3166	3124	3103	3080
80	3026	2957	2854	2798	2728	2675	2619	2506	2468	2449	2437	2427	2416	2403	2384	2371
96	2362	2351	2339	2328	2314	2301	2292	2281	2268	2254	2240	2224	2211	2197	2182	2157
112	2135	2114	2090	2075	2060	2046	2031	2017	2003	1990	1979	1971	1959	1947	1935	1923
128	1911	1902	1889	1869	1857	1846	1835	1823	1805	1796	1789	1781	1773	1768	1763	1757
144	1751	1742	1735	1728	1720	1709	1703	1695	1687	1680	1674	1666	1659	1653	1647	1638
160	1630	1622	1612	1604	1596	1590	1582	1576	1568	1558	1551	1545	1538	1531	1523	1517
176	1509	1497	1488	1476	1467	1457	1447	1437	1430	1423	1413	1402	1392	1383	1370	1361
192	1350	1343	1336	1329	1322	1315	1310	1303	1293	1285	1279	1270	1263	1255	1247	1238
208	1230	1222	1213	1207	1201	1194	1186	1180	1173	1166	1158	1152	1141	1133	1127	1120
224	1112	1105	1098	1093	1088	1079	1071	1064	1055	1049	1045	1041	1035	1031	1026	1021
240	1017	1008	1002	998	993	985	981	973	968	964	956	951	946	942	935	930
256	924	917	912	906	900	895	889	884	880	876	871	865	860	854	848	843
272	836	830	825	819	815	809	804	796	792	788	779	775	766	761	758	751
288	743	739	734	729	726	720	716	712	706	703	702	701	700	698	695	692
304	691	689	687	683	680	677	673	671	669	668	665	662	660	659	657	656
320	653	643	631	622	613	604	597	589	582	573	565	558	551	540	531	519
336	508	490	483	477	468	461	452	444	437	427	418	408	402	391	385	378
352	362	359	356	353	351	349	347	344	342	340	337	335	333	330	328	325
368	322	319	316	314	312	310	308	306	304	302	299	297	295	292	290	287
384	284	277	275	273	270	268	265	263	261	258	255	252	251	247	245	243
400	237	235	232	230	227	226	224	222	220	218	216	214	212	210	207	204
416	202	199	197	195	193	191	190	188	186	184	182	180	178	176	175	173
432	170	168	167	166	165	164	163	162	161	160	159	158	157	156	155	153
448	151	148	146	145	143	142	141	139	138	136	134	132	131	129	128	126
464	123	121	120	119	118	117	116	115	114	113	112	111	110	108	107	105
480	103	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87
496	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70
512	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54
528	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38
544	0															

FIG 16(27)

pki=27

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	42239	39798	37437	35071	32585	30088	27716	25355	22919	22401	21937	21489	21025	20534	19875	19379
16	18902	18403	17776	17232	16550	15866	15325	14692	14192	13713	13217	12564	12073	11605	11151	10684
32	10168	9993	9855	9701	9537	9397	9234	9064	8914	8689	8518	8357	8194	8011	7844	7608
48	7438	7267	7028	6859	6677	6514	6356	6187	5960	5811	5643	5479	5338	5174	5021	4886
64	4712	4670	4571	4512	4450	4381	4320	4206	4132	4075	4041	4004	3952	3899	3861	3825
80	3765	3691	3576	3513	3445	3383	3324	3224	3181	3159	3145	3134	3122	3107	3079	3061
96	3048	3035	3022	3007	2986	2969	2957	2943	2922	2904	2885	2866	2845	2824	2800	2770
112	2735	2701	2671	2647	2626	2603	2584	2567	2550	2530	2516	2504	2487	2464	2450	2436
128	2423	2410	2392	2365	2350	2338	2326	2311	2290	2281	2273	2265	2255	2249	2243	2236
144	2229	2217	2209	2201	2190	2179	2172	2163	2151	2142	2135	2126	2116	2107	2097	2088
160	2078	2068	2058	2048	2039	2029	2020	2013	2004	1990	1981	1972	1960	1951	1941	1932
176	1922	1908	1895	1881	1868	1853	1840	1826	1813	1799	1784	1771	1756	1743	1729	1716
192	1703	1693	1683	1673	1664	1651	1642	1633	1619	1609	1602	1593	1584	1575	1565	1554
208	1543	1533	1524	1515	1507	1497	1488	1481	1473	1460	1451	1443	1431	1420	1412	1403
224	1393	1386	1378	1372	1366	1356	1349	1342	1333	1327	1323	1318	1312	1308	1302	1296
240	1291	1282	1275	1270	1263	1253	1247	1238	1232	1227	1219	1212	1205	1199	1190	1184
256	1175	1168	1163	1156	1149	1142	1134	1128	1123	1118	1112	1103	1096	1089	1082	1077
272	1069	1061	1055	1047	1041	1034	1027	1020	1015	1009	1000	994	985	979	974	967
288	958	953	947	941	937	931	926	921	916	912	910	908	906	904	900	896
304	894	892	888	883	877	871	865	861	858	856	852	848	846	844	843	842
320	838	827	813	800	789	778	768	757	749	737	726	716	708	694	680	664
336	652	634	625	617	607	597	585	576	567	553	542	528	520	505	494	485
352	469	465	460	455	451	448	444	440	437	434	430	425	422	418	413	408
368	404	401	397	394	391	389	386	383	380	377	374	371	369	366	363	360
384	357	351	349	347	344	341	338	335	333	329	326	323	321	317	314	311
400	305	302	298	294	291	289	286	283	281	279	276	273	271	268	264	260
416	257	254	252	250	247	245	243	241	239	236	233	231	229	226	224	222
432	219	217	216	215	213	212	211	209	208	207	205	203	202	200	199	197
448	195	192	190	188	186	184	182	180	178	176	174	172	170	167	165	162
464	158	156	155	154	153	152	151	149	148	146	144	142	141	139	137	135
480	133	131	130	129	128	127	126	125	124	123	122	121	120	118	117	116
496	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99
512	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82
528	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66
544	0															

1628

FIG 16(28)

pki=28

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	52866	50598	48703	46846	44739	42635	40780	38893	36621	35963	35375	34775	34162	33551	32762	32191
16	31607	31006	30317	29680	28912	28143	27508	26821	26221	25642	25069	24284	23675	23061	22461	21872
32	21214	20908	20666	20394	20135	19893	19619	19321	19048	18673	18378	18106	17849	17547	17289	16917
48	16656	16395	16026	15768	15469	15211	14937	14639	14264	13990	13693	13417	13175	12917	12648	12408
64	12104	12020	11797	11672	11539	11391	11257	10991	10837	10713	10660	10605	10506	10407	10351	10297
80	10172	10016	9752	9620	9474	9341	9214	8994	8908	8857	8821	8789	8757	8719	8664	8628
96	8600	8563	8526	8494	8455	8421	8393	8362	8324	8289	8246	8203	8164	8126	8084	8021
112	7953	7886	7823	7780	7742	7703	7659	7615	7580	7542	7511	7484	7451	7411	7379	7341
128	7304	7276	7241	7186	7149	7117	7086	7050	6999	6971	6948	6923	6896	6877	6857	6833
144	6811	6780	6754	6729	6702	6672	6652	6628	6604	6582	6560	6533	6509	6488	6465	6438
160	6409	6378	6347	6321	6296	6272	6245	6222	6198	6164	6139	6118	6093	6072	6049	6025
176	6000	5968	5941	5903	5871	5831	5800	5764	5738	5712	5676	5644	5605	5573	5535	5509
192	5476	5452	5428	5405	5383	5358	5336	5311	5277	5253	5230	5203	5178	5153	5127	5096
208	5065	5036	5009	4986	4965	4941	4914	4892	4869	4845	4822	4802	4772	4745	4721	4695
224	4664	4641	4617	4597	4578	4552	4527	4504	4476	4454	4439	4423	4403	4388	4365	4345
240	4329	4304	4284	4269	4252	4225	4209	4181	4163	4145	4121	4100	4082	4066	4038	4019
256	3990	3966	3948	3929	3909	3888	3861	3843	3828	3813	3795	3769	3748	3727	3708	3691
272	3666	3637	3619	3591	3575	3557	3536	3512	3495	3477	3448	3432	3405	3387	3372	3352
288	3328	3313	3292	3270	3255	3235	3219	3204	3183	3173	3168	3163	3158	3151	3141	3133
304	3128	3121	3113	3100	3087	3074	3061	3053	3046	3041	3033	3023	3016	3011	3006	3001
320	2991	2952	2908	2866	2830	2790	2751	2714	2682	2641	2603	2564	2532	2488	2444	2397
336	2356	2290	2259	2228	2192	2151	2098	2060	2027	1980	1940	1884	1851	1803	1768	1735
352	1675	1663	1648	1635	1623	1612	1598	1586	1575	1564	1551	1537	1526	1514	1501	1486
368	1474	1462	1450	1439	1428	1419	1407	1395	1385	1375	1364	1352	1342	1331	1320	1308
384	1296	1271	1263	1254	1243	1232	1216	1205	1197	1183	1171	1153	1145	1129	1120	1111
400	1089	1080	1066	1055	1045	1037	1026	1016	1008	1000	990	980	972	962	951	937
416	928	916	907	899	890	882	872	863	854	845	836	825	817	807	799	790
432	778	771	766	762	758	755	750	746	743	738	733	725	721	715	710	705
448	696	684	677	670	662	656	647	639	632	624	615	605	598	589	581	571
464	557	551	547	543	538	535	530	525	520	515	509	502	497	490	484	476
480	466	459	456	453	449	446	442	438	434	430	426	421	418	413	409	404
496	395	393	392	391	389	388	386	384	381	379	377	374	372	369	367	364
512	358	357	356	355	354	353	352	351	349	348	347	346	345	344	343	341
528	337	336	335	334	333	332	331	330	329	328	327	326	325	324	323	321
544	0															

1629

FIG 16(29)

1630

pki=29

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	60575	59072	57887	56796	55638	54830	54146	53415	52337	51692	50981	50391	50031	49682	49181	48677
16	48030	47693	47299	46778	46301	45864	45536	45139	44810	44516	44222	43637	43259	42882	42567	42245
32	41843	41472	41123	40904	40676	40340	40071	39826	39628	39277	39027	38821	38613	38282	38060	37738
48	37538	37322	37063	36869	36677	36423	36125	35904	35625	35397	35180	34976	34804	34587	34383	34209
64	33992	33807	33502	33351	33176	33033	32859	32612	32451	32325	32234	32147	32013	31897	31829	31759
80	31596	31433	31181	31050	30902	30768	30644	30469	30308	30171	30038	29947	29885	29809	29680	29602
96	29535	29464	29408	29361	29291	29221	29154	29098	29027	28961	28862	28804	28742	28670	28591	28491
112	28383	28272	28192	28122	28067	27998	27935	27880	27833	27768	27719	27678	27629	27544	27490	27435
128	27387	27346	27288	27153	27081	27014	26960	26896	26793	26689	26611	26551	26486	26417	26365	26305
144	26254	26136	26063	26009	25945	25879	25817	25769	25709	25656	25600	25557	25511	25451	25388	25325
160	25265	25202	25155	25109	25071	25011	24958	24919	24877	24810	24763	24723	24674	24619	24568	24507
176	24456	24396	24340	24264	24210	24114	24062	23999	23949	23892	23825	23769	23709	23653	23604	23557
192	23512	23465	23423	23380	23344	23284	23217	23166	23096	23045	22998	22956	22914	22867	22818	22766
208	22718	22669	22629	22589	22553	22502	22457	22422	22387	22335	22294	22258	22209	22145	22075	22021
224	21951	21900	21853	21810	21772	21713	21655	21610	21544	21457	21414	21362	21309	21249	21180	21124
240	21084	21022	20975	20926	20884	20826	20780	20717	20677	20624	20582	20536	20500	20452	20379	20333
256	20280	20230	20195	20154	20115	20069	20017	19976	19944	19896	19851	19774	19728	19680	19636	19595
272	19552	19491	19456	19405	19371	19329	19283	19237	19202	19165	19124	19092	19051	19014	18979	18941
288	18898	18854	18805	18746	18709	18659	18621	18583	18531	18492	18474	18455	18440	18420	18383	18355
304	18338	18319	18294	18260	18223	18187	18159	18134	18115	18103	18085	18045	18024	18009	17996	17983
320	17953	17838	17717	17581	17458	17336	17222	17099	16979	16852	16727	16581	16455	16285	16120	15958
336	15776	15590	15479	15362	15226	15075	14888	14749	14614	14437	14285	14045	13882	13701	13530	13377
352	13116	13066	13009	12945	12887	12841	12783	12729	12668	12617	12559	12474	12401	12344	12264	12189
368	12098	12050	11999	11946	11890	11843	11783	11729	11672	11625	11568	11503	11442	11385	11316	11251
384	11173	11087	11045	11001	10948	10890	10816	10761	10717	10645	10583	10493	10436	10356	10290	10228
400	10122	10069	10005	9933	9870	9823	9765	9702	9641	9591	9526	9452	9388	9321	9241	9163
416	9085	9025	8972	8918	8855	8800	8728	8666	8599	8546	8481	8407	8340	8277	8210	8143
432	8056	8018	7992	7963	7932	7911	7879	7847	7820	7790	7760	7702	7666	7625	7584	7546
448	7475	7407	7352	7294	7224	7174	7097	7029	6958	6902	6831	6736	6654	6580	6492	6404
464	6275	6234	6196	6157	6104	6072	6017	5965	5910	5870	5818	5748	5685	5629	5560	5487
480	5382	5334	5303	5273	5226	5199	5147	5104	5059	5027	4984	4927	4881	4833	4787	4733
496	4641	4622	4609	4594	4566	4552	4522	4496	4463	4449	4424	4391	4361	4332	4301	4262
512	4193	4189	4186	4181	4168	4163	4149	4137	4119	4115	4104	4088	4074	4060	4046	4025
528	3973	3972	3971	3968	3967	3964	3962	3956	3955	3952	3948	3945	3942	3939	3933	3895
544	0															

FIG 16(30)

1631

pki = 30

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	65535	65534	65533	65532	65531	58892	56860	55734	54374	54373	54372	54371	52965	51873	51071	51070
16	51069	49113	48101	48100	45626	44234	43070	42303	41309	41267	40913	40210	39534	39108	39085	39059
32	38918	38917	38916	37703	36985	36984	35687	34958	34629	34070	33564	33546	33425	33424	31910	30975
48	30726	30149	30126	29864	29584	29138	28779	28762	28748	28394	28377	28361	28349	28218	28109	28096
64	28049	27785	27784	27001	26498	26307	25906	25867	25742	25719	25461	25093	23893	23568	23519	23498
80	23241	23212	23190	23184	23082	23076	23056	22973	22954	22728	22558	22452	22448	22419	22354	22102
96	21829	21342	21282	21256	21227	20959	20797	20737	20668	20273	19177	19069	19030	18658	18542	17235
112	17082	16853	16602	16548	16517	16447	16333	16327	16309	16272	16255	16251	16242	16207	16173	16167
128	16163	16162	16155	16136	16124	16118	16117	16116	16109	15967	15818	15807	15778	15721	15712	15686
144	15679	15649	15624	15622	15613	15354	15195	15139	15060	14766	14473	14408	14382	14123	14005	13482
160	13395	13281	13230	13199	13178	13121	13077	13072	13060	13044	13028	13026	13010	12838	12782	12729
176	12696	12304	12187	11023	10941	10705	10642	10607	10580	10500	10482	10379	10376	10297	10222	10206
192	10173	10153	10151	10139	10137	10095	10056	10047	10036	9998	9990	9988	9987	9951	9944	9925
208	9922	9920	9916	9913	9910	9886	9868	9866	9865	9849	9841	9840	9833	9823	9815	9814
224	9813	9807	9806	9803	9802	9793	9789	9786	9783	9677	9667	9640	9630	9612	9608	9590
240	9589	9440	9355	9306	9286	8915	8814	7809	7752	7687	7646	7630	7614	7556	7509	7465
256	7459	7425	7418	7392	7383	7370	7368	7355	7352	7311	7300	7289	7286	7260	7245	7239
272	7237	7229	7227	7224	7222	7211	7208	7184	7183	7176	7165	7162	7153	7147	7146	7139
288	7137	7133	7132	7131	7130	7125	7124	7123	7120	7092	7089	7084	7083	7077	7069	7009
304	7001	6989	6979	6850	6827	6807	6783	6772	6766	6765	6764	6761	6758	6757	6756	6755
320	6754	6738	6658	6624	6577	6528	6520	6466	6408	6331	6300	6251	6191	6147	6093	5840
336	5639	5559	5543	5529	5503	5455	5439	5406	5316	5258	5219	5080	4980	4836	4775	4710
352	4125	4122	4098	4066	4034	4030	4022	3982	3939	3932	3904	3839	3770	3753	3716	3577
368	3441	3439	3436	3430	3409	3408	3400	3388	3355	3353	3332	3321	3295	3286	3257	3235
384	3106	3076	3072	3068	3055	3039	3038	3030	3014	2992	2981	2973	2964	2915	2902	2882
400	2757	2750	2711	2694	2680	2678	2675	2638	2604	2601	2585	2565	2542	2531	2501	2410
416	2345	2337	2329	2327	2311	2308	2297	2291	2252	2246	2233	2230	2212	2206	2183	2172
432	2013	2005	1999	1993	1988	1986	1985	1977	1972	1966	1958	1914	1902	1883	1875	1860
448	1828	1817	1810	1800	1787	1781	1771	1758	1725	1714	1692	1669	1640	1632	1611	1560
464	1442	1439	1433	1427	1417	1415	1408	1398	1376	1365	1354	1338	1325	1310	1291	1260
480	1178	1167	1164	1163	1159	1157	1153	1148	1129	1121	1117	1112	1109	1099	1095	1078
496	1003	1001	1000	999	995	994	991	988	980	979	978	976	970	968	962	956
512	900	899	898	897	895	894	893	890	885	884	882	880	877	874	873	869
528	833	832	831	830	829	828	827	825	824	823	822	821	820	819	818	808
544	0															

FIG 16(31)

pki = 31

index i	val[i+0]	val[i+1]	val[i+2]	val[i+3]	val[i+4]	val[i+5]	val[i+6]	val[i+7]	val[i+8]	val[i+9]	val[i+10]	val[i+11]	val[i+12]	val[i+13]	val[i+14]	val[i+15]
0	65535	65534	65533	65532	65531	63229	61960	60737	58893	58892	58891	58890	57935	57060	55961	55960
16	55959	54884	53760	53759	52324	51190	50660	49534	48642	48552	48216	47048	46052	45248	45174	45102
32	44760	44759	44758	43863	43139	43138	42018	41334	41037	40112	39441	39390	39218	39217	38238	37256
48	37024	36270	36212	36031	35783	34919	33995	33940	33893	33128	33070	33029	32995	32789	32548	32518
64	32351	31839	31838	31166	30541	30350	29760	29669	29515	29461	29068	28651	27968	27829	27797	27768
80	27187	27082	27006	26989	26828	26812	26759	26604	26546	26091	25432	25107	25082	25019	24886	24513
96	24160	23756	23710	23691	23653	23259	22907	22840	22750	22327	21403	21354	21316	20938	20820	20210
112	20135	19966	19881	19838	19817	19714	19620	19611	19595	19554	19536	19533	19522	19432	19360	19334
128	19329	19326	19311	19277	19248	19226	19224	19220	19195	18732	18390	18331	18263	17993	17970	17911
144	17889	17758	17685	17677	17646	17268	16892	16817	16716	16299	15953	15919	15897	15517	15384	14975
160	14918	14810	14772	14752	14736	14633	14554	14549	14535	14510	14492	14489	14475	14069	13993	13891
176	13856	13471	13364	12685	12644	12413	12379	12354	12332	12225	12195	12105	12100	12050	12016	11998
192	11970	11949	11946	11933	11928	11843	11736	11713	11688	11617	11589	11584	11581	11511	11487	11422
208	11419	11411	11406	11403	11400	11363	11327	11325	11319	11299	11286	11284	11266	11242	11212	11210
224	11208	11194	11192	11190	11188	11167	11157	11154	11148	10721	10698	10639	10600	10535	10523	10496
240	10489	10146	10057	9974	9940	9568	9444	8806	8773	8674	8644	8627	8614	8540	8446	8379
256	8372	8332	8325	8310	8295	8281	8277	8264	8261	8155	8129	8064	8053	7985	7954	7929
272	7923	7903	7899	7896	7893	7862	7851	7803	7802	7789	7773	7769	7753	7742	7739	7724
288	7715	7696	7693	7691	7690	7679	7676	7673	7670	7604	7592	7571	7569	7563	7547	7468
304	7451	7443	7435	7345	7326	7306	7293	7283	7277	7276	7270	7264	7258	7257	7256	7255
320	7250	7231	7185	7155	7100	7068	7059	7001	6925	6877	6829	6753	6631	6555	6470	6187
336	5866	5807	5787	5769	5719	5692	5650	5614	5505	5453	5401	5265	5151	5057	4974	4871
352	4213	4208	4192	4173	4148	4142	4132	4109	4081	4072	4045	3990	3912	3894	3853	3736
368	3612	3608	3598	3591	3571	3566	3555	3542	3509	3504	3486	3470	3440	3424	3391	3345
384	3203	3178	3174	3168	3153	3146	3139	3130	3107	3093	3081	3067	3043	3003	2984	2955
400	2706	2700	2674	2661	2642	2639	2633	2610	2589	2583	2565	2541	2511	2492	2466	2395
416	2334	2322	2313	2308	2290	2285	2273	2263	2228	2219	2203	2192	2165	2148	2127	2104
432	2000	1996	1992	1988	1981	1979	1978	1971	1963	1957	1948	1922	1909	1897	1887	1867
448	1766	1752	1742	1735	1719	1714	1705	1692	1665	1655	1638	1614	1580	1563	1540	1499
464	1386	1380	1374	1370	1359	1356	1348	1337	1319	1312	1301	1289	1271	1256	1236	1202
480	1126	1114	1111	1108	1100	1098	1093	1086	1072	1068	1062	1054	1045	1034	1023	1004
496	928	926	925	923	918	917	915	910	899	897	893	888	881	875	870	857
512	813	812	811	810	808	807	806	804	798	797	796	794	791	789	786	780
528	754	753	752	751	750	749	748	746	745	744	743	742	741	740	738	723
544	0															

1632

FIG 16(32)

1710 dgroups[]

1712a

1712b

index i (1720)	val[i+0] (1722)	val[i+1] (1724)	val[i+2] (1726)	val[i+3] (1728)	val[i+4] (1730)	val[i+5] (1732)	val[i+6] (1734)	val[i+7]
0	0x000001	0x000101	0x000201	0x000301	0x000401	0x000501	0x000601	0x000701
8	0x000801	0x000901	0x000A01	0x000B01	0x000C01	0x000D01	0x000E01	0x000F01
16	0x001001	0x001101	0x001201	0x001301	0x001401	0x001501	0x001601	0x001701
24	0x001801	0x001901	0x001A01	0x001B01	0x001C01	0x001D01	0x001E01	0x001F01
32	0x002001	0x002101	0x002201	0x002301	0x002401	0x002501	0x002601	0x002701
40	0x002801	0x002901	0x002A01	0x002B01	0x002C01	0x002D01	0x002E01	0x002F01
48	0x003001	0x003101	0x003201	0x003301	0x003401	0x003501	0x003601	0x003701
56	0x003801	0x003901	0x003A01	0x003B01	0x003C01	0x003D01	0x003E01	0x003F01
64	0x004001	0x004101	0x004201	0x004301	0x004401	0x004501	0x004601	0x004701
72	0x004801	0x004901	0x004A01	0x004B01	0x004C01	0x004D01	0x004E01	0x004F01
80	0x005001	0x005101	0x005201	0x005301	0x005401	0x005501	0x005601	0x005701
88	0x005801	0x005901	0x005A01	0x005B01	0x005C01	0x005D01	0x005E01	0x005F01
96	0x006001	0x006101	0x006201	0x006301	0x006401	0x006501	0x006601	0x006701
104	0x006801	0x006901	0x006A01	0x006B01	0x006C01	0x006D01	0x006E01	0x006F01
112	0x007001	0x007101	0x007201	0x007301	0x007401	0x007501	0x007601	0x007701
120	0x007801	0x007901	0x007A01	0x007B01	0x007C01	0x007D01	0x007E01	0x007F01
128	0x008001	0x008101	0x008201	0x008301	0x008401	0x008501	0x008601	0x008701
136	0x008801	0x008901	0x008A01	0x008B01	0x008C01	0x008D01	0x008E01	0x008F01
144	0x009001	0x009101	0x009201	0x009301	0x009401	0x009501	0x009601	0x009701
152	0x009801	0x009901	0x009A01	0x009B01	0x009C01	0x009D01	0x009E01	0x009F01
160	0x00A001	0x00A101	0x00A201	0x00A301	0x00A401	0x00A501	0x00A601	0x00A701
168	0x00A801	0x00A901	0x00AA01	0x00AB01	0x00AC01	0x00AD01	0x00AE01	0x00AF01
176	0x00B001	0x00B101	0x00B201	0x00B301	0x00B401	0x00B501	0x00B601	0x00B701
184	0x00B801	0x00B901	0x00BA01	0x00BB01	0x00BC01	0x00BD01	0x00BE01	0x00BF01
192	0x00C001	0x00C101	0x00C201	0x00C301	0x00C401	0x00C501	0x00C601	0x00C701
200	0x00C801	0x00C901	0x00CA01	0x00CB01	0x00CC01	0x00CD01	0x00CE01	0x00CF01
208	0x00D001	0x00D101	0x00D201	0x00D301	0x00D401	0x00D501	0x00D601	0x00D701
216	0x00D801	0x00D901	0x00DA01	0x00DB01	0x00DC01	0x00DD01	0x00DE01	0x00DF01
224	0x00E001	0x00E101	0x00E201	0x00E301	0x00E401	0x00E501	0x00E601	0x00E701
232	0x00E801	0x00E901	0x00EA01	0x00EB01	0x00EC01	0x00ED01	0x00EE01	0x00EF01
240	0x00F001	0x00F101	0x00F201	0x00F301	0x00F401	0x00F501	0x00F601	0x00F701
248	0x00F801	0x00F901	0x00FA01	0x00FB01	0x00FC01	0x00FD01	0x00FE01	0x00FF01
256	0x010001	0x010101	0x010201	0x010301	0x010401	0x010501	0x010601	0x010701

FIG 17(1)

1710	1720	1722	1724	1726	1728	1730	1732	1734
264	0x010801	0x010901	0x010A01	0x010B01	0x010C01	0x010D01	0x010E01	0x010F01
272	0x011001	0x011101	0x011201	0x011301	0x011401	0x011501	0x011601	0x011701
280	0x011801	0x011901	0x011A01	0x011B01	0x011C01	0x011D01	0x011E01	0x011F01
288	0x012001	0x012101	0x012201	0x012301	0x012401	0x012501	0x012601	0x012701
296	0x012801	0x012901	0x012A01	0x012B01	0x012C01	0x012D01	0x012E01	0x012F01
304	0x013001	0x013101	0x013201	0x013301	0x013401	0x013501	0x013601	0x013701
312	0x013801	0x013901	0x013A01	0x013B01	0x013C01	0x013D01	0x013E01	0x013F01
320	0x014001	0x014107	0x014806	0x014E06	0x015406	0x015A07	0x016106	0x016706
328	0x016D06	0x017307	0x017A06	0x018006	0x018606	0x018C08	0x019407	0x019B07
336	0x01A207	0x01A90A	0x01B308	0x01BB08	0x01C308	0x01CB0A	0x01D508	0x01DD08
344	0x01E508	0x01ED0B	0x01F809	0x020109	0x020A09	0x02130B	0x021E09	0x022709
352	0x023009	0x023906	0x023F06	0x024506	0x024B06	0x025106	0x025706	0x025D06
360	0x026306	0x026906	0x026F06	0x027506	0x027B06	0x028106	0x028706	0x028D06
368	0x029306	0x029906	0x029F06	0x02A506	0x02AB06	0x02B106	0x02B706	0x02BD06
376	0x02C306	0x02C906	0x02CF06	0x02D506	0x02DB06	0x02E106	0x02E706	0x02ED06
384	0x02F306	0x02F908	0x030106	0x030706	0x030D06	0x031308	0x031B06	0x032106
392	0x032706	0x032D09	0x033607	0x033D07	0x034407	0x034B09	0x035407	0x035B07
400	0x036207	0x03690C	0x03750C	0x03810C	0x038D0C	0x03990C	0x03A50C	0x03B10C
408	0x03BD0C	0x03C90C	0x03D50C	0x03E10C	0x03ED0C	0x03F90C	0x04050C	0x04110C
416	0x041D0C	0x04290C	0x04350C	0x04410C	0x044D0C	0x04590C	0x04650C	0x04710C
424	0x047D0C	0x04890C	0x04950C	0x04A10C	0x04AD0C	0x04B90C	0x04C50C	0x04D10C
432	0x04DD0C	0x04E907	0x04F007	0x04F708	0x04FF08	0x050708	0x050F07	0x051609
440	0x051F09	0x052809	0x053108	0x05390A	0x05430A	0x054D0B	0x055809	0x05610A
448	0x056B0A	0x057515	0x058A15	0x059F17	0x05B618	0x05CE16	0x05E417	0x05FB18
456	0x06131A	0x062D17	0x064418	0x065C19	0x06751B	0x069019	0x06A91B	0x06C41B
464	0x06DF1E	0x06FD10	0x070D15	0x072218	0x073A1D	0x075716	0x076D1A	0x07871E
472	0x07A522	0x07C719	0x07E01D	0x07FD20	0x081D26	0x08431F	0x086223	0x088525
480	0x08AA2B	0x08D512	0x08E719	0x09001D	0x091D23	0x094019	0x09591E	0x097722
488	0x099929	0x09C21D	0x09DF22	0x0A0126	0x0A272C	0x0A5324	0x0A7728	0x0A9F2C
496	0x0ACB32	0x0AFD0A	0x0B0716	0x0B1D18	0x0B3525	0x0B5A16	0x0B7022	0x0B9224
504	0x0BB630	0x0BE618	0x0BFE24	0x0C2227	0x0C4933	0x0C7C25	0x0CA130	0x0CD133
512	0x0D043F	0x0D4301	0x0D440B	0x0D4F0C	0x0D5B1B	0x0D760B	0x0D8118	0x0D991B
520	0x0DB42D	0x0DE10C	0x0DED1B	0x0E081E	0x0E2632	0x0E581B	0x0E732D	0x0EA032
528	0x0ED249	0x0F1B01	0x0F1C01	0x0F1D09	0x0F2601	0x0F2708	0x0F2F09	0x0F381A
536	0x0F5201	0x0F5309	0x0F5C0A	0x0F661D	0x0F8309	0x0F8C1A	0x0FA61D	0x0FC33D

FIG 17(2)

dgvectors[]

1810

1812a

1812b

index i	1820	1822	1824	1826	1828	1830	1832	1834	1836	1838	1840	1842	1844	1846	1848	1850	1852	1854	1856	1858	1860	1862	1864	1866	1868	1870	1872	1874	1876	1878	1880	1882
0	0	0	0	0	-1	0	0	0	0	-1	0	0	0	0	-1	0	0	0	0	-1	0	0	0	1	0	0	1	0	0	1	0	0
32	1	0	0	0	-1	-1	0	0	-1	0	-1	0	-1	0	0	-1	-1	0	0	1	-1	0	1	0	-1	1	0	0	0	-1	-1	0
64	0	-1	0	-1	0	-1	0	1	0	-1	1	0	0	0	-1	-1	0	0	-1	1	0	0	1	-1	0	0	1	1	0	1	-1	0
96	0	1	0	-1	0	1	0	1	0	1	1	0	1	-1	0	0	1	0	-1	0	1	0	0	-1	1	0	0	1	1	0	1	0
128	1	1	0	0	-1	-1	-1	0	-1	-1	0	-1	-1	-1	0	1	-1	-1	1	0	-1	0	-1	-1	-1	0	-1	1	-1	0	1	-1
160	-1	0	1	1	-1	1	-1	0	-1	1	0	-1	-1	1	0	1	-1	1	1	0	0	-1	-1	-1	0	-1	-1	1	0	-1	1	-1
192	0	-1	1	1	0	1	-1	-1	0	1	-1	1	0	1	1	-1	0	1	1	1	1	-1	-1	0	1	-1	0	-1	1	-1	0	1
224	1	-1	1	0	1	0	-1	-1	1	0	-1	1	1	0	1	-1	1	0	1	1	1	1	-1	0	1	1	0	-1	1	1	0	1
256	1	1	1	0	-2	0	0	0	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	1	1	-1	1	-1	-1	-1	1	-1	1
288	-1	1	1	-1	-1	1	1	1	0	-2	0	0	0	0	-2	0	0	0	0	-2	0	0	0	2	0	0	2	0	0	2	0	0
320	1	-1	-1	-1	1	-1	-1	1	1	-1	1	-1	1	-1	1	1	1	1	-1	-1	1	1	-1	1	1	1	1	-1	1	1	1	1
352	2	0	0	0	-2	-1	0	0	-2	0	-1	0	-2	0	0	-1	-2	0	0	1	-2	0	1	0	-2	1	0	0	-1	-2	0	0
384	-1	0	-2	0	-1	0	0	-2	-1	0	0	2	-1	0	2	0	-1	2	0	0	0	-2	-1	0	0	-2	0	-1	0	-2	0	1
416	0	-2	1	0	0	-1	-2	0	0	-1	0	-2	0	-1	0	2	0	-1	2	0	0	0	-2	-1	0	0	-2	1	0	0	-1	-2
448	0	0	-1	2	0	0	1	-2	0	0	1	2	0	0	2	-1	0	0	2	1	0	1	-2	0	0	1	0	-2	0	1	0	2
480	0	1	2	0	0	2	-1	0	0	2	0	-1	0	2	0	1	0	2	1	0	1	-2	0	0	1	0	-2	0	1	0	0	-2
512	1	0	0	2	1	0	2	0	1	2	0	0	2	-1	0	0	2	0	-1	0	2	0	0	-1	2	0	0	1	2	0	1	0
544	2	1	0	0	-2	-1	-1	0	-2	-1	0	-1	-2	-1	0	1	-2	-1	1	0	-2	0	-1	-1	-2	0	-1	1	-2	0	1	-1
576	-2	0	1	1	-2	1	-1	0	-2	1	0	-1	-2	1	0	1	-2	1	1	0	-1	-2	-1	0	-1	-2	0	-1	-1	-2	0	1
608	-1	-2	1	0	-1	-1	-2	0	-1	-1	0	-2	-1	-1	0	2	-1	-1	2	0	-1	0	-2	-1	-1	0	-2	1	-1	0	-1	-2
640	-1	0	-1	2	-1	0	1	-2	-1	0	1	2	-1	0	2	-1	-1	0	2	1	-1	1	-2	0	-1	1	0	-2	-1	1	0	2
672	-1	1	2	0	-1	2	-1	0	-1	2	0	-1	-1	2	0	1	-1	2	1	0	0	-2	-1	-1	0	-2	-1	1	0	-2	1	-1
704	0	-2	1	1	0	-1	-2	-1	0	-1	-2	1	0	-1	-1	-2	0	-1	-1	2	0	-1	1	-2	0	-1	1	2	0	-1	2	-1
736	0	-1	2	1	0	1	-2	-1	0	1	-2	1	0	1	-1	-2	0	1	-1	2	0	1	1	-2	0	1	1	2	0	1	2	-1
768	0	1	2	1	0	2	-1	-1	0	2	-1	1	0	2	1	-1	0	2	1	1	1	-2	-1	0	1	-2	0	-1	1	-2	0	1
800	1	-2	1	0	1	-1	-2	0	1	-1	0	-2	1	-1	0	2	1	-1	2	0	1	0	-2	-1	1	0	-2	1	1	0	-1	-2
832	1	0	-1	2	1	0	1	-2	1	0	1	2	1	0	2	-1	1	0	2	1	1	1	-2	0	1	1	0	-2	1	1	0	2
864	1	1	2	0	1	2	-1	0	1	2	0	-1	1	2	0	1	1	2	1	0	2	-1	-1	0	2	-1	0	-1	2	-1	0	1
896	2	-1	1	0	2	0	-1	-1	2	0	-1	1	2	0	1	-1	2	0	1	1	2	1	-1	0	2	1	0	-1	2	1	0	1
928	2	1	1	0	-2	-1	-1	-1	-2	-1	-1	1	-2	-1	1	-1	-2	-1	1	1	-2	1	-1	-1	-2	1	-1	1	-2	1	1	-1
960	-2	1	1	1	-1	-2	-1	-1	-1	-2	-1	1	-1	-2	1	-1	-1	-2	1	1	-1	-1	-2	-1	-1	-1	-2	1	-1	-1	-1	-2
992	-1	-1	-1	2	-1	-1	1	-2	-1	-1	1	2	-1	-1	2	-1	-1	-1	2	1	-1	1	-2	-1	-1	1	-2	1	-1	1	-1	-2
1024	-1	1	-1	2	-1	1	1	-2	-1	1	1	2	-1	1	2	-1	-1	1	2	1	-1	2	-1	-1	-1	2	-1	1	-1	2	1	-1
1056	-1	2	1	1	1	-2	-1	-1	1	-2	-1	1	1	-2	1	-1	1	-2	1	1	1	-1	-2	-1	1	-1	-2	1	1	-1	-1	-2
1088	1	-1	-1	2	1	-1	1	-2	1	-1	1	2	1	-1	2	-1	1	-1	2	1	1	1	-2	-1	1	1	-2	1	1	1	-1	-2
1120	1	1	-1	2	1	1	1	-2	1	1	1	2	1	1	2	-1	1	1	2	1	1	2	-1	-1	1	2	-1	1	1	2	1	-1
1152	1	2	1	1	2	-1	-1	-1	2	-1	-1	1	2	-1	1	-1	2	-1	1	1	2	1	-1	-1	2	1	-1	1	2	1	1	-1
1184	2	1	1	1	-2	-2	0	0	-2	0	-2	0	-2	0	0	-2	-2	0	0	2	-2	0	2	0	-2	2	0	0	0	-2	-2	0
1216	0	-2	0	-2	0	-2	0	2	0	-2	2	0	0	0	-2	-2	0	0	-2	2	0	0	2	-2	0	0	2	2	0	2	-2	0
1248	0	2	0	-2	0	2	0	2	0	2	2	0	2	-2	0	0	2	0	-2	0	2	0	0	-2	2	0	0	2	2	0	2	0
1280	2	2	0	0	0	0	3	0	2	2	1	0	1	2	2	0	1	0	2	2	2	1	2	0	2	0	2	1	2	0	1	2
1312	0	2	1	2	-2	2	1	0	0	1	2	2	-1	2	2	0	-2	1	2	0	0	2	2	1	0	-2	2	1	1	-2	2	0
1344	0	-1	2	2	0	-2	1	2	2	-2	1	0	2	-1	2	0	-1	-2	2	0	-2	-1	2	0	-2	0	2	1	-2	-2	1	0
1376	-2	0	1	2	-1	0	2	2	2	2	0	1	0	0	0	3	2	0	-2	1	2	1	0	2	1	0	-2	2	2	0	-1	2
1408	1	2	0	2	0	2	-1	2	0	2	-2	1	0	1	-2	2	-2	1	0	2	-1	2	0	2	-2	2	0	1	2	-1	0	2
1440	0	-1	-2	2	2	-2	0	1	0	-2	-2	1	1	-2	0	2	0	-2	-1	2	-1	0	-2	2	-1	-2	0	2	-2	0	-2	1
1472	-2	0	-1	2	-2	-1	0	2	-2	-2	0	1	0	0	0	-3	2	2	0	-1	2	1	0	-2	2	0	2	-1	2	0	1	-2

FIG 18(1)

1504	1	2	0	-2	1	0	2	-2	0	2	1	-2	0	1	2	-2	-2	2	0	-1	-2	1	0	-2	-1	2	0	-2	0	2	2	-1
1536	0	-1	2	-2	0	-2	2	-1	0	-2	1	-2	2	-1	0	-2	2	-2	0	-1	1	-2	0	-2	-2	-1	0	-2	-2	0	1	-2
1568	-1	-2	0	-2	-2	-2	0	-1	-2	0	2	-1	-1	0	2	-2	3	0	0	0	0	0	-3	0	2	2	-1	0	2	1	-2	0
1600	2	0	-1	-2	2	0	-2	-1	1	2	-2	0	1	0	-2	-2	0	2	-1	-2	0	2	-2	-1	0	1	-2	-2	-2	2	-1	0
1632	-2	1	-2	0	-1	2	-2	0	0	3	0	0	0	-1	-2	-2	2	-1	-2	0	0	-2	-1	-2	2	-2	-1	0	0	-2	-2	-1
1664	0	-3	0	0	1	-2	-2	0	-1	0	-2	-2	-1	-2	-2	0	-2	0	-1	-2	-2	-1	-2	0	-2	0	-2	-1	-2	-2	-1	0
1696	-3	0	0	0	1	2	1	2	0	0	3	1	2	1	2	1	1	2	2	1	1	0	3	0	0	0	1	3	2	1	1	2
1728	3	0	1	0	2	2	1	1	1	1	2	2	-2	1	2	1	-1	2	1	2	-1	1	2	2	-2	1	1	2	0	1	3	0
1760	-1	2	2	1	-2	2	1	1	0	3	1	0	1	-2	2	1	2	-1	1	2	0	-1	3	0	2	-1	2	1	1	-2	1	2
1792	0	-3	1	0	1	-1	2	2	2	-2	1	1	-2	-1	1	2	-1	-1	2	2	-1	0	3	0	-2	-2	1	1	-1	-2	2	1
1824	-3	0	1	0	-2	-1	2	1	-1	-2	1	2	1	2	-1	2	1	0	0	3	2	2	-1	1	2	1	-2	1	1	1	-2	2
1856	0	0	-3	1	3	0	0	1	0	0	-1	3	1	2	-2	1	2	1	-1	2	-1	2	-1	2	-1	2	-2	1	-2	1	-1	2
1888	0	3	0	1	-2	1	-2	1	0	1	0	3	-2	2	-1	1	-1	1	-2	2	0	-3	0	1	2	-2	-1	1	2	-1	-1	2
1920	2	-1	-2	1	1	-1	-2	2	0	-1	0	3	1	-2	-1	2	1	-2	-2	1	-2	-1	-1	2	-2	-1	-2	1	-1	0	0	3
1952	-1	-2	-1	2	-1	-1	-2	2	-3	0	0	1	-1	-2	-2	1	-2	-2	-1	1	1	2	2	-1	1	2	1	-2	1	1	2	-2
1984	1	0	0	-3	0	0	3	-1	0	0	1	-3	3	1	0	0	3	0	0	-1	2	2	1	-1	2	1	2	-1	2	1	1	-2
2016	-1	3	0	0	-2	2	1	-1	-1	2	2	-1	-1	2	1	-2	-2	1	2	-1	-1	1	2	-2	-2	1	1	-2	0	3	0	-1
2048	0	1	0	-3	0	-3	0	-1	2	-1	1	-2	2	-2	1	-1	1	-1	2	-2	1	-2	2	-1	1	-2	1	-2	2	-1	2	-1
2080	3	-1	0	0	0	-1	0	-3	-1	-3	0	0	-2	-1	1	-2	-3	0	0	-1	-1	0	0	-3	-1	-1	2	-2	-1	-2	2	-1
2112	-2	-2	1	-1	-1	-2	1	-2	-2	-1	2	-1	2	1	-2	-1	1	3	0	0	1	2	-1	-2	1	2	-2	-1	1	1	-2	-2
2144	1	0	-3	0	0	0	-1	-3	3	0	-1	0	0	0	-3	-1	2	2	-1	-1	2	1	-1	-2	-1	2	-1	-2	-2	2	-1	-1
2176	-1	2	-2	-1	-3	1	0	0	-1	1	-2	-2	-2	1	-1	-2	0	3	-1	0	0	1	-3	0	-2	1	-2	-1	0	-3	-1	0
2208	2	-1	-1	-2	2	-1	-2	-1	2	-2	-1	-1	1	-1	-2	-2	1	-2	-1	-2	1	-2	-2	-1	1	-3	0	0	0	-1	-3	0
2240	-2	-2	-1	-1	-2	-1	-1	-2	-1	0	-3	0	-2	-1	-2	-1	-1	-1	-2	-2	-1	-2	-1	-2	-1	-2	-2	-1	-3	0	-1	0
2272	-3	-1	0	0	1	0	3	1	3	0	1	1	1	3	1	0	1	0	1	3	3	1	1	0	1	1	3	0	-3	1	1	0
2304	0	1	1	3	0	3	1	1	-1	1	3	0	-1	3	1	0	0	1	3	1	1	-1	3	0	0	-1	3	1	0	-1	1	3
2336	1	-3	1	0	3	-1	1	0	0	-3	1	1	-1	0	3	1	-1	-1	3	0	-1	0	1	3	-3	0	1	1	-3	-1	1	0
2368	-1	-3	1	0	3	0	-1	1	1	3	0	1	3	1	0	1	1	0	-1	3	1	1	0	3	1	0	-3	1	0	3	-1	1
2400	-1	1	0	3	0	1	-1	3	0	1	-3	1	-3	1	0	1	-1	3	0	1	1	-3	0	1	0	-3	-1	1	0	-1	-3	1
2432	0	-1	-1	3	1	-1	0	3	3	-1	0	1	-1	-3	0	1	-3	0	-1	1	-3	-1	0	1	-1	0	-1	3	-1	0	-3	1
2464	-1	-1	0	3	3	0	1	-1	3	1	0	-1	1	3	0	-1	1	1	0	-3	1	0	3	-1	1	0	1	-3	-1	1	0	-3
2496	0	3	1	-1	0	1	3	-1	0	1	1	-3	-3	1	0	-1	-1	3	0	-1	1	-3	0	-1	0	-1	3	-1	0	-1	1	-3
2528	3	-1	0	-1	0	-3	1	-1	1	-1	0	-3	-3	0	1	-1	-1	-3	0	-1	-1	0	3	-1	-1	0	1	-3	-1	-1	0	-3
2560	-3	-1	0	-1	3	1	-1	0	3	0	-1	-1	1	3	-1	0	1	1	-3	0	1	0	-1	-3	1	0	-3	-1	-1	1	-3	0
2592	0	3	-1	-1	0	1	-1	-3	0	1	-3	-1	-3	1	-1	0	-1	3	-1	0	1	-3	-1	0	0	-1	-1	-3	3	-1	-1	0
2624	0	-1	-3	-1	0	-3	-1	-1	1	-1	-3	0	-1	-3	-1	0	-1	0	-1	-3	-1	0	-3	-1	-3	0	-1	-1	-1	-1	-3	0
2656	-3	-1	-1	0	2	0	2	2	1	1	3	1	3	1	1	1	2	2	2	0	1	1	1	3	1	3	1	1	0	2	2	2
2688	-1	1	3	1	-1	3	1	1	-3	1	1	1	-2	2	2	0	-1	1	1	3	1	-3	1	1	3	-1	1	1	0	-2	2	2
2720	2	-2	2	0	1	-1	3	1	1	-1	1	3	-1	-1	1	3	-2	0	2	2	-1	-3	1	1	-3	-1	1	1	-2	-2	2	0
2752	-1	-1	3	1	3	1	-1	1	1	1	-1	3	1	3	-1	1	2	0	-2	2	2	2	0	2	1	1	-3	1	0	2	-2	2
2784	-3	1	-1	1	-1	1	-1	3	-2	2	0	2	-1	1	-3	1	-1	3	-1	1	1	-1	-1	3	1	-1	-3	1	0	-2	-2	2
2816	1	-3	-1	1	3	-1	-1	1	2	-2	0	2	-1	-1	-1	3	-2	0	-2	2	-1	-1	-3	1	-2	-2	0	2	-1	-3	-1	1
2848	-3	-1	-1	1	2	2	0	-2	2	0	2	-2	1	3	1	-1	1	1	3	-1	3	1	1	-1	1	1	1	-3	-3	1	1	-1
2880	0	2	2	-2	-1	3	1	-1	-1	1	3	-1	-1	1	1	-3	-2	2	0	-2	1	-1	3	-1	1	-1	1	-3	1	-3	1	-1
2912	0	-2	2	-2	3	-1	1	-1	2	-2	0	-2	-1	-1	3	-1	-2	0	2	-2	-1	-1	1	-3	-3	-1	1	-1	-1	-3	1	-1
2944	-2	-2	0	-2	3	1	-1	-1	2	2	-2	0	2	0	-2	-2	1	3	-1	-1	1	1	-1	-3	1	1	-3	-1	0	2	-2	-2
2976	-1	3	-1	-1	-3	1	-1	-1	-1	1	-1	-3	-2	2	-2	0	-1	1	-3	-1	1	-1	-1	-3	1	-1	-3	-1	1	-3	-1	-1
3008	0	-2	-2	-2	3	-1	-1	-1	2	-2	-2	0	-1	-1	-1	-3	-2	0	-2	-2	-1	-1	-3	-1	-1	-3	-1	-1	-2	-2	-2	0
3040	-3	-1	-1	-1	0	0	3	2	2	0	3	0	2	2	2	1	2	1	2	2	3	0	2	0	0	0	2	3	2	2	1	2
3072	1	2	2	2	-2	2	2	1	-1	2	2	2	0	3	2	0	-2	2	1	2	0	2	3	0	-2	1	2	2	2	-2	2	1
3104	2	-1	2	2	1	-2	2	2	0	-3	2	0	2	-2	1	2	0	-2	3	0	-2	-2	2	1	-2	0	3	0	-2	-2	1	2

FIG 18(2)

3136	-1	-2	2	2	-3	0	2	0	-2	-1	2	2	0	0	-2	3	0	0	-3	2	2	2	-2	1	1	2	-2	2	2	2	-1	2
3168	2	1	-2	2	3	0	0	2	2	0	0	3	0	2	0	3	-2	2	-1	2	-2	2	-2	1	-1	2	-2	2	-2	1	-2	2
3200	0	3	0	2	2	-1	-2	2	0	-3	0	2	0	-2	0	3	2	-2	-1	2	2	-2	-2	1	1	-2	-2	2	-2	0	0	3
3232	-2	-2	-1	2	-1	-2	-2	2	-2	-2	-2	1	-2	-1	-2	2	-3	0	0	2	0	0	3	-2	0	0	2	-3	3	2	0	0
3264	3	0	0	-2	2	2	2	-1	2	2	1	-2	2	1	2	-2	2	0	0	-3	1	2	2	-2	0	2	0	-3	-1	2	2	-2
3296	-2	1	2	-2	-2	3	0	0	-2	2	2	-1	0	3	0	-2	-2	2	1	-2	3	-2	0	0	0	-2	0	-3	0	-3	0	-2
3328	2	-1	2	-2	2	-2	2	-1	2	-2	1	-2	1	-2	2	-2	-2	-2	2	-1	-2	-3	0	0	-2	-2	1	-2	-3	0	0	-2
3360	-2	0	0	-3	-1	-2	2	-2	-2	-1	2	-2	0	0	-2	-3	3	0	-2	0	2	3	0	0	0	0	-3	-2	2	2	-1	-2
3392	2	2	-2	-1	2	1	-2	-2	2	0	-3	0	1	2	-2	-2	0	2	-3	0	-2	2	-1	-2	-2	2	-2	-1	-2	1	-2	-2
3424	-3	2	0	0	-1	2	-2	-2	0	3	-2	0	0	-2	-3	0	0	-3	-2	0	2	-1	-2	-2	2	-2	-1	-2	2	-2	-2	-1
3456	2	-3	0	0	1	-2	-2	-2	-2	0	-3	0	-2	-2	-1	-2	-1	-2	-2	-2	-2	-1	-2	-2	-2	-2	-2	-1	-3	0	-2	0
3488	-3	-2	0	0	1	3	2	0	3	0	2	1	1	0	2	3	1	0	3	2	3	1	2	0	3	0	1	2	2	0	3	1
3520	2	3	1	0	3	2	1	0	2	1	3	0	2	0	1	3	1	2	3	0	0	1	3	2	0	3	1	2	-3	1	2	0
3552	0	2	1	3	-1	2	3	0	-2	1	3	0	0	1	2	3	0	3	2	1	0	2	3	1	-3	2	1	0	-2	3	1	0
3584	-1	3	2	0	3	-2	1	0	1	-3	2	0	2	-3	1	0	0	-1	3	2	0	-3	2	1	0	-2	3	1	3	-1	2	0
3616	2	-1	3	0	1	-2	3	0	0	-2	1	3	0	-1	2	3	0	-3	1	2	-2	-3	1	0	-3	0	1	2	-1	-3	2	0
3648	-1	-2	3	0	-2	0	1	3	-1	0	3	2	-1	0	2	3	-2	0	3	1	-3	0	2	1	-2	-1	3	0	-3	-2	1	0
3680	-3	-1	2	0	3	0	-2	1	3	1	0	2	1	2	0	3	3	2	0	1	2	3	0	1	2	0	-3	1	2	1	0	3
3712	3	0	-1	2	1	3	0	2	1	0	-2	3	2	0	-1	3	1	0	-3	2	0	3	-2	1	0	2	-1	3	0	2	-3	1
3744	0	1	-2	3	0	1	-3	2	-3	2	0	1	-2	3	0	1	-2	1	0	3	-3	1	0	2	-1	3	0	2	0	3	-1	2
3776	-1	2	0	3	1	-2	0	3	1	-3	0	2	0	-1	-2	3	3	-1	0	2	2	-1	0	3	3	-2	0	1	0	-2	-3	1
3808	0	-3	-1	2	0	-1	-3	2	0	-3	-2	1	2	-3	0	1	0	-2	-1	3	-2	-1	0	3	-1	0	-2	3	-3	-1	0	2
3840	-1	0	-3	2	-1	-2	0	3	-2	-3	0	1	-1	-3	0	2	-3	-2	0	1	-3	0	-1	2	-3	0	-2	1	-2	0	-1	3
3872	-2	0	-3	1	3	0	2	-1	2	0	3	-1	3	1	0	-2	3	2	0	-1	3	0	1	-2	2	3	0	-1	2	1	0	-3
3904	2	0	1	-3	1	3	0	-2	1	2	0	-3	1	0	3	-2	1	0	2	-3	0	3	2	-1	0	3	1	-2	0	2	3	-1
3936	0	2	1	-3	0	1	3	-2	0	1	2	-3	-2	3	0	-1	-3	2	0	-1	-2	1	0	-3	-3	1	0	-2	-1	3	0	-2
3968	-1	2	0	-3	1	-2	0	-3	1	-3	0	-2	0	-1	3	-2	0	-1	2	-3	3	-1	0	-2	0	-2	3	-1	3	-2	0	-1
4000	0	-2	1	-3	0	-3	2	-1	0	-3	1	-2	2	-1	0	-3	2	-3	0	-1	-1	0	3	-2	-1	0	2	-3	-2	-1	0	-3
4032	-3	0	2	-1	-1	-2	0	-3	-2	-3	0	-1	-3	0	1	-2	-1	-3	0	-2	-3	-1	0	-2	-2	0	3	-1	-2	0	1	-3
4064	-3	-2	0	-1	3	2	-1	0	3	1	-2	0	3	0	-1	-2	3	0	-2	-1	2	3	-1	0	2	1	-3	0	2	0	-1	-3
4096	2	0	-3	-1	1	3	-2	0	1	2	-3	0	1	0	-2	-3	1	0	-3	-2	0	3	-1	-2	0	3	-2	-1	0	2	-1	-3
4128	0	2	-3	-1	0	1	-2	-3	0	1	-3	-2	-2	3	-1	0	-3	2	-1	0	-2	1	-3	0	-3	1	-2	0	-1	3	-2	0
4160	-1	2	-3	0	1	-2	-3	0	1	-3	-2	0	0	-1	-2	-3	0	-1	-3	-2	3	-1	-2	0	3	-2	-1	0	0	-2	-1	-3
4192	0	-2	-3	-1	0	-3	-1	-2	2	-1	-3	0	0	-3	-2	-1	2	-3	-1	0	-1	0	-2	-3	-2	-1	-3	0	-1	0	-3	-2
4224	-1	-2	-3	0	-2	-3	-1	0	-1	-3	-2	0	-3	0	-1	-2	-3	-1	-2	0	-3	0	-2	-1	-2	0	-1	-3	-2	0	-3	-1
4256	-3	-2	-1	0	1	3	2	1	1	2	1	3	1	2	3	1	1	1	2	3	1	1	3	2	2	1	3	1	2	1	1	3
4288	3	1	2	1	3	2	1	1	3	1	1	2	2	3	1	1	1	3	1	2	-2	3	1	1	-2	1	1	3	-1	2	3	1
4320	-2	1	3	1	-1	3	2	1	-1	1	2	3	-3	2	1	1	-1	2	1	3	-1	1	3	2	-3	1	2	1	-3	1	1	2
4352	-1	3	1	2	1	-2	3	1	3	-2	1	1	1	-2	1	3	1	-3	2	1	1	-1	2	3	3	-1	1	2	2	-1	1	3
4384	2	-1	3	1	1	-1	3	2	1	-3	1	2	3	-1	2	1	2	-3	1	1	-2	-1	1	3	-1	-3	1	2	-3	-1	2	1
4416	-1	-3	2	1	-3	-2	1	1	-1	-2	3	1	-1	-2	1	3	-1	-1	2	3	-2	-1	3	1	-1	-1	3	2	-3	-1	1	2
4448	-2	-3	1	1	1	1	-2	3	1	1	-3	2	2	1	-3	1	3	1	-1	2	3	1	-2	1	1	2	-3	1	2	3	-1	1
4480	2	1	-1	3	3	2	-1	1	1	3	-1	2	1	3	-2	1	1	2	-1	3	-2	3	-1	1	-1	3	-1	2	-3	1	-2	1
4512	-1	3	-2	1	-2	1	-1	3	-2	1	-3	1	-1	2	-1	3	-1	2	-3	1	-3	2	-1	1	-1	1	-3	2	-1	1	-2	3
4544	-3	1	-1	2	1	-1	-2	3	1	-1	-3	2	1	-2	-1	3	3	-1	-2	1	3	-2	-1	1	1	-3	-1	2	1	-3	-2	1
4576	3	-1	-1	2	1	-2	-3	1	2	-1	-1	3	2	-1	-3	1	2	-3	-1	1	-1	-2	-1	3	-2	-1	-3	1	-1	-2	-3	1
4608	-1	-3	-1	2	-2	-3	-1	1	-3	-1	-2	1	-1	-3	-2	1	-1	-1	-2	3	-3	-1	-1	2	-1	-1	-3	2	-2	-1	-1	3
4640	-3	-2	-1	1	1	1	3	-2	1	1	2	-3	3	1	2	-1	3	1	1	-2	3	2	1	-1	2	3	1	-1	2	1	3	-1
4672	2	1	1	-3	1	3	2	-1	1	3	1	-2	1	2	3	-1	1	2	1	-3	-1	3	2	-1	-2	3	1	-1	-3	2	1	-1
4704	-1	3	1	-2	-1	2	3	-1	-2	1	3	-1	-1	2	1	-3	-2	1	1	-3	-1	1	3	-2	-3	1	2	-1	-1	1	2	-3
4736	-3	1	1	-2	1	-1	3	-2	1	-1	2	-3	1	-2	3	-1	1	-2	1	-3	1	-3	2	-1	1	-3	1	-2	3	-1	2	-1

FIG 18(3)

4768	3	-1	1	-2	3	-2	1	-1	2	-1	3	-1	2	-1	1	-3	2	-3	1	-1	-1	-2	3	-1	-1	-2	1	-3	-3	-1	2	-1
4800	-1	-3	2	-1	-2	-3	1	-1	-1	-3	1	-2	-3	-2	1	-1	-3	-1	1	-2	-1	-1	3	-2	-2	-1	3	-1	-1	-1	2	-3
4832	-2	-1	1	-3	1	1	-2	-3	1	1	-3	-2	3	2	-1	-1	3	1	-1	-2	3	1	-2	-1	2	3	-1	-1	2	1	-1	-3
4864	2	1	-3	-1	1	3	-1	-2	1	3	-2	-1	1	2	-1	-3	1	2	-3	-1	-3	1	-1	-2	-3	1	-2	-1	-1	3	-1	-2
4896	-2	3	-1	-1	-1	3	-2	-1	-3	2	-1	-1	-1	2	-1	-3	-1	2	-3	-1	-2	1	-1	-3	-1	1	-2	-3	-2	1	-3	-1
4928	-1	1	-3	-2	1	-1	-2	-3	1	-1	-3	-2	1	-2	-1	-3	1	-2	-3	-1	1	-3	-1	-2	1	-3	-2	-1	3	-1	-1	-2
4960	3	-1	-2	-1	3	-2	-1	-1	2	-1	-1	-3	2	-1	-3	-1	2	-3	-1	-1	-2	-1	-1	-3	-1	-2	-1	-3	-2	-1	-3	-1
4992	-1	-2	-3	-1	-3	-1	-1	-2	-1	-3	-1	-2	-2	-3	-1	-1	-1	-3	-2	-1	-3	-1	-2	-1	-1	-1	-2	-3	-1	-1	-3	-2
5024	-3	-2	-1	-1	2	0	2	3	3	2	2	0	2	2	2	2	2	2	3	0	2	0	3	2	2	3	2	0	3	0	2	2
5056	-3	2	2	0	-2	2	3	0	0	3	2	2	-2	2	2	2	0	2	3	2	0	2	2	3	-2	3	2	0	0	-2	2	3
5088	3	-2	2	0	0	-4	1	0	0	-3	2	2	2	-2	2	2	0	-2	3	2	2	-2	3	0	2	-3	2	0	-2	0	2	3
5120	-4	0	1	0	-3	0	2	2	-2	-3	2	0	-3	-2	2	0	-2	0	3	2	-2	-2	2	2	-2	-2	3	0	2	2	-2	2
5152	3	0	-2	2	2	0	-3	2	2	2	0	3	2	3	0	2	0	0	-4	1	2	0	-2	3	3	2	0	2	-2	2	-2	2
5184	-2	2	0	3	-2	3	0	2	0	2	-2	3	0	3	-2	2	0	2	-3	2	-3	2	0	2	2	-2	-2	2	2	-2	0	3
5216	2	-3	0	2	1	-4	0	0	0	-2	-2	3	0	-2	-3	2	0	-3	-2	2	3	-2	0	2	0	-4	0	1	-2	-2	-2	2
5248	-2	0	-2	3	-4	0	0	1	-2	0	-3	2	-2	-2	0	3	-1	-4	0	0	-3	-2	0	2	-2	-3	0	2	-3	0	-2	2
5280	2	3	0	-2	2	2	0	-3	2	0	3	-2	2	0	2	-3	1	0	0	-4	0	0	1	-4	2	2	2	-2	3	0	2	-2
5312	3	2	0	-2	-2	3	0	-2	-2	2	0	-3	0	3	2	-2	0	2	3	-2	0	2	2	-3	0	1	0	-4	-2	2	2	-2
5344	-3	2	0	-2	2	-2	2	-2	2	-2	0	-3	2	-3	0	-2	0	0	0	-4	0	-1	0	-4	0	-2	3	-2	0	-2	2	-3
5376	0	-3	2	-2	0	-4	0	-1	3	-2	0	-2	-3	-2	0	-2	-2	0	3	-2	-2	0	2	-3	-4	0	0	-1	-2	-2	0	-3
5408	-4	-1	0	0	-2	-3	0	-2	-3	0	2	-2	-2	-2	2	-2	-1	0	0	-4	2	3	-2	0	2	2	-3	0	2	0	-2	-3
5440	2	0	-3	-2	1	0	-4	0	0	0	-4	0	0	0	-1	-4	0	0	-4	-1	2	2	-2	-2	3	2	-2	0	3	0	-2	-2
5472	-2	3	-2	0	-2	2	-3	0	-4	1	0	0	0	3	-2	-2	-2	2	-2	-2	0	2	-2	-3	0	2	-3	-2	0	1	-4	0
5504	-3	2	-2	0	2	-2	-3	0	2	-2	-2	-2	2	-3	-2	0	0	-4	0	0	0	-1	-4	0	0	-2	-2	-3	0	-2	-3	-2
5536	0	-3	-2	-2	0	-4	-1	0	3	-2	-2	0	-3	0	-2	-2	-3	-2	-2	0	-2	0	-2	-3	-2	0	-3	-2	-4	0	-1	0
5568	-2	-2	-3	0	-2	-3	-2	0	-1	0	-4	0	-2	-2	-2	-2	-4	0	0	0	3	1	2	2	3	0	3	0	3	2	2	1
5600	1	2	3	2	1	3	3	0	2	2	3	1	1	2	2	3	2	2	1	3	3	1	3	0	0	0	3	3	2	3	1	2
5632	3	2	1	2	3	0	1	3	1	0	3	3	2	1	3	2	1	3	2	2	2	1	2	3	3	3	1	0	3	0	3	1
5664	3	3	0	0	2	3	2	1	-3	2	1	2	-2	2	1	3	-2	3	1	2	-3	1	2	2	-2	3	2	1	-2	2	3	1
5696	-1	3	2	2	-4	1	1	1	0	3	3	0	0	3	3	1	-3	3	1	0	-3	2	2	1	-1	3	3	0	-2	1	3	2
5728	-2	1	2	3	-4	1	1	0	-1	2	3	2	-1	2	2	3	-3	1	3	0	0	3	1	3	0	1	3	3	1	-4	1	1
5760	3	-3	1	0	3	-1	2	2	3	-2	2	1	1	-3	2	2	2	-3	1	2	3	-2	1	2	0	-3	3	1	2	-2	3	1
5792	1	-4	1	0	2	-2	1	3	2	-3	2	1	0	-3	3	0	2	-1	3	2	3	-1	3	0	1	-2	3	2	1	-2	2	3
5824	0	-3	1	3	2	-1	2	3	1	-3	3	0	0	-4	1	1	0	-1	3	3	3	-3	0	0	-2	-2	3	1	-2	-2	1	3
5856	-2	-1	3	2	-1	-2	3	2	-1	-2	2	3	-3	-1	3	0	-3	-1	2	2	-4	-1	1	1	-2	-3	1	2	-3	0	3	0
5888	-3	0	1	3	-1	-3	3	0	-2	-3	2	1	-3	-2	2	1	-1	-4	1	0	-3	-2	1	2	-2	-1	2	3	-3	-3	1	0
5920	-1	-4	1	1	-3	0	3	1	-1	-3	2	2	-4	0	1	1	-1	0	3	3	-4	-1	1	0	3	2	-1	2	2	3	-1	2
5952	2	3	-2	1	2	2	-1	3	1	3	-2	2	2	1	-3	2	1	3	0	3	2	2	-3	1	1	2	-3	2	3	0	0	3
5984	1	0	-3	3	3	1	-2	2	1	1	-4	1	3	1	0	3	3	3	0	1	1	0	-4	1	3	0	-1	3	3	0	-3	1
6016	0	0	-3	3	3	2	-2	1	2	1	-2	3	1	2	-2	3	-3	1	-2	2	0	3	-1	3	-2	2	-1	3	-1	2	-2	3
6048	0	1	-3	3	-1	2	-3	2	0	3	-3	1	-2	1	-3	2	-2	3	-2	1	0	3	0	3	-3	2	-1	2	-1	3	-2	2
6080	-1	3	0	3	-1	1	-4	1	-2	2	-3	1	-2	3	-1	2	-3	1	0	3	0	1	-4	1	-3	2	-2	1	-2	1	-2	3
6112	-4	1	0	1	-3	3	0	1	-4	1	-1	1	3	-2	-1	2	3	-2	-2	1	2	-1	-2	3	2	-2	-1	3	2	-2	-3	1
6144	3	-1	0	3	3	-1	-2	2	1	-3	0	3	1	-1	-4	1	1	-4	-1	1	1	-4	0	1	0	-1	-3	3	2	-3	-2	1
6176	1	-2	-3	2	3	-3	0	1	1	-3	-2	2	1	-2	-2	3	0	-3	-1	3	2	-1	-3	2	0	-1	-4	1	0	-3	0	3
6208	2	-3	-1	2	0	-3	-3	1	0	-4	-1	1	-2	-1	-2	3	-3	-2	-2	1	-2	-1	-3	2	-3	0	0	3	-1	0	-4	1
6240	-2	-2	-1	3	-2	-2	-3	1	-1	-2	-2	3	-4	0	-1	1	-1	-2	-3	2	-4	-1	0	1	-1	-3	-2	2	-3	-1	-2	2
6272	-1	-4	0	1	-2	-3	-1	2	-2	-3	-2	1	-1	-1	-4	1	-1	0	-3	3	-1	-3	0	3	-1	-4	-1	1	-3	0	-1	3
6304	-3	0	-3	1	-3	-1	0	3	-3	-2	-1	2	-3	-3	0	1	-4	-1	-1	1	3	1	2	-2	3	0	0	-3	1	3	0	-3
6336	2	3	2	-1	2	3	1	-2	2	2	3	-1	2	2	1	-3	2	1	3	-2	1	3	2	-2	2	1	2	-3	3	1	0	-3
6368	3	0	1	-3	1	1	0	-4	1	1	1	-4	1	2	3	-2	1	2	2	-3	1	0	1	-4	0	0	3	-3	3	3	0	-1

FIG 18(4)

6400	3	0	3	-1	3	2	2	-1	3	2	1	-2	1	0	3	-3	-1	3	2	-2	-1	2	3	-2	-1	2	2	-3	-1	1	0	-4
6432	0	3	3	-1	-4	1	0	-1	-2	3	1	-2	0	3	1	-3	-1	3	0	-3	0	1	3	-3	-2	3	2	-1	-2	2	3	-1
6464	-3	2	2	-1	-2	2	1	-3	-1	1	1	-4	-3	2	1	-2	0	3	0	-3	-3	3	0	-1	0	1	1	-4	-2	1	3	-2
6496	-3	1	0	-3	-2	1	2	-3	-4	1	1	-1	-3	1	2	-2	3	-1	2	-2	3	-2	2	-1	3	-2	1	-2	2	-1	3	-2
6528	2	-1	2	-3	2	-2	3	-1	2	-2	1	-3	2	-3	2	-1	2	-3	1	-2	3	-1	0	-3	3	-3	0	-1	1	-1	1	-4
6560	1	-3	0	-3	1	-4	1	-1	1	-1	0	-4	0	-1	3	-3	1	-2	3	-2	0	-3	3	-1	1	-2	2	-3	0	-3	1	-3
6592	1	-3	2	-2	1	-4	0	-1	0	-1	1	-4	0	-3	0	-3	0	-4	1	-1	-2	-1	2	-3	-4	0	1	-1	-2	-2	3	-1
6624	-1	0	1	-4	-2	-2	1	-3	-3	0	0	-3	-1	-1	0	-4	-1	-2	3	-2	-1	-2	2	-3	-2	-3	2	-1	-3	-1	2	-2
6656	-1	-3	2	-2	-2	-3	1	-2	-1	-4	0	-1	-3	-2	2	-1	-1	0	3	-3	-1	-1	1	-4	-1	-3	0	-3	-3	-2	1	-2
6688	-1	-4	1	-1	-3	0	3	-1	-3	0	1	-3	-3	-1	0	-3	-3	-3	0	-1	-4	-1	0	-1	-4	-1	1	-1	-2	-1	3	-2
6720	3	1	-2	-2	3	0	-3	0	2	3	-1	-2	2	3	-2	-1	2	2	-1	-3	2	2	-3	-1	2	1	-2	-3	2	1	-3	-2
6752	3	3	-1	0	3	1	-3	0	3	0	-1	-3	3	0	-3	-1	1	3	-2	-2	1	3	-3	0	1	1	-1	-4	1	1	-4	-1
6784	1	0	-3	-3	1	2	-2	-3	1	2	-3	-2	1	1	-4	0	1	0	-1	-4	1	0	-4	-1	0	0	-3	-3	3	2	-1	-2
6816	3	2	-2	-1	-1	3	-2	-2	-3	1	-2	-2	-1	2	-2	-3	-1	2	-3	-2	-1	1	-4	0	-2	3	-1	-2	0	3	-1	-3
6848	0	3	-3	-1	-3	3	0	0	-2	3	-2	-1	0	1	-3	-3	-1	3	-3	0	-1	1	-1	-4	-1	1	-4	-1	-2	2	-1	-3
6880	-2	2	-3	-1	-4	1	-1	0	0	3	-3	0	-3	3	-1	0	0	1	-1	-4	-3	1	-3	0	-3	2	-1	-2	0	1	-4	-1
6912	-2	1	-2	-3	-3	2	-2	-1	-2	1	-3	-2	-4	1	-1	-1	0	-4	-1	-1	3	-1	-2	-2	3	-2	-1	-2	3	-2	-2	-1
6944	2	-1	-2	-3	2	-1	-3	-2	2	-2	-1	-3	2	-2	-3	-1	2	-3	-1	-2	2	-3	-2	-1	3	-1	-3	0	3	-3	-1	0
6976	1	-1	-1	-4	1	-1	-4	-1	1	-3	-3	0	1	-4	-1	-1	1	-1	-4	0	0	-1	-3	-3	0	-3	-1	-3	1	-2	-2	-3
7008	0	-3	-3	-1	1	-2	-3	-2	1	-3	-2	-2	1	-4	-1	0	0	-1	-1	-4	0	-1	-4	-1	0	-3	-3	0	-2	-1	-2	-3
7040	-2	-1	-3	-2	-3	-2	-2	-1	-1	0	-1	-4	-1	0	-4	-1	-1	-1	-4	0	-2	-2	-1	-3	-3	0	-3	0	-3	-3	0	0
7072	-2	-2	-3	-1	-1	-2	-2	-3	-1	-2	-3	-2	-1	-3	-2	-2	-1	-4	-1	0	-2	-3	-1	-2	-3	-1	-2	-2	-2	-3	-2	-1
7104	-4	0	-1	-1	-1	0	-3	-3	-1	-1	-1	-4	-1	-1	-4	-1	-1	-3	-3	0	-1	-4	-1	-1	-3	0	-1	-3	-3	0	-3	-1
7136	-3	-1	-3	0	-3	-3	-1	0	-4	-1	-1	-1	-3	-2	-1	-2	-4	-1	-1	0	3	2	3	0	3	0	2	3	3	0	3	2
7168	1	3	3	1	3	1	1	3	2	2	2	3	3	3	1	1	2	3	2	2	2	3	3	0	3	1	3	1	2	0	3	3
7200	3	3	2	0	1	3	1	3	2	2	3	2	3	2	2	2	1	1	3	3	-2	3	2	2	-2	3	3	0	-3	1	1	3
7232	-3	2	3	0	-1	3	1	3	0	3	2	3	-1	3	3	1	-4	2	1	1	-4	1	2	0	-3	2	2	2	0	2	3	3
7264	-3	3	2	0	0	3	3	2	-2	2	3	2	-3	1	3	1	-1	1	3	3	-4	1	1	2	-4	2	1	0	-2	2	2	3
7296	-3	3	1	1	-4	1	2	1	2	-3	3	0	2	-3	2	2	2	-4	1	0	1	-1	3	3	0	-2	3	3	1	-3	1	3
7328	3	-3	2	0	3	-3	1	1	3	-2	2	2	0	-4	2	0	1	-4	2	1	0	-3	2	3	2	-2	2	3	3	-2	3	0
7360	0	-4	1	2	1	-4	1	2	0	-4	2	1	0	-3	3	2	3	-1	1	3	2	-2	3	2	2	-4	1	1	1	-4	2	0
7392	3	-1	3	1	1	-3	3	1	-4	0	1	2	-2	-3	3	0	-4	0	2	0	-3	-1	1	3	-2	-2	2	3	-2	-2	3	2
7424	-3	-1	3	1	-2	-4	1	0	-2	0	3	3	-1	-4	2	1	-2	-4	1	1	-4	-1	2	1	-1	-3	3	1	-1	-4	1	2
7456	-4	-1	1	2	-1	-3	1	3	-4	-2	1	0	-2	-3	2	2	-1	-4	2	0	-3	0	3	2	-4	-2	1	1	-3	0	2	3
7488	-3	-2	3	0	-1	-1	3	3	-3	-3	2	0	-3	-2	2	2	-3	-3	1	1	-4	-1	2	0	-4	0	2	1	2	2	-3	2
7520	1	0	-4	2	3	2	-2	2	1	1	-4	2	1	2	-4	1	3	3	-1	1	2	3	-2	2	2	2	-2	3	1	3	-1	3
7552	1	3	-3	1	3	1	-1	3	1	1	-3	3	3	3	0	2	3	0	-2	3	0	0	-4	2	2	0	-3	3	3	2	0	3
7584	3	0	-3	2	2	3	0	3	2	1	-4	1	3	1	-3	1	2	0	-4	1	-3	3	-1	1	0	3	-2	3	-3	1	-1	3
7616	-2	2	-3	2	-2	2	-2	3	-2	1	-4	1	-1	1	-4	2	0	3	-3	2	0	2	-4	1	-1	2	-4	1	-1	1	-3	3
7648	-2	3	-2	2	-4	2	-1	1	-1	3	-3	1	-2	3	0	3	-3	3	0	2	-3	2	0	3	0	1	-4	2	-3	2	-2	2
7680	-4	1	-1	2	-4	1	-2	1	0	2	-3	3	-4	2	0	1	-4	1	0	2	-3	1	-3	1	-1	3	-1	3	2	-1	-4	1
7712	2	-2	-2	3	0	-2	-4	1	2	-4	-1	1	2	-4	0	0	0	-4	0	2	2	-3	-2	2	2	-3	0	3	1	-4	-2	1
7744	3	-3	-1	1	0	-4	-2	1	1	-4	-1	2	1	-1	-4	2	3	-2	0	3	1	-2	-4	1	0	-4	-1	2	3	-1	-3	1
7776	2	-4	0	1	1	-4	0	2	3	-1	-1	3	2	-2	-3	2	0	-3	-3	2	0	-3	-2	3	0	-2	-3	3	3	-3	0	2
7808	1	-1	-3	3	1	-3	-1	3	0	-1	-4	2	1	-3	-3	1	3	-2	-2	2	-4	-1	0	2	-4	-2	0	1	-3	-1	-1	3
7840	-1	-2	-4	1	-2	0	-4	1	-1	0	-4	2	-1	-1	-4	2	-1	-4	-1	2	-1	-4	-2	1	-1	-1	-3	3	-1	-4	0	2
7872	-2	0	-3	3	-2	-1	-4	1	-2	-3	0	3	-2	-4	-1	1	-2	-2	-2	3	-2	-2	-3	2	-3	0	-2	3	-2	-3	-2	2
7904	-3	0	-3	2	-3	-2	0	3	-2	-4	0	1	-3	-3	0	2	-1	-3	-1	3	-3	-2	-2	2	-4	-1	-1	2	-4	-1	-2	1
7936	-4	-2	-1	1	-1	-3	-3	1	-4	0	-1	2	-3	-1	-3	1	-4	0	-2	1	-3	-3	-1	1	-4	0	0	2	3	3	1	-1
7968	1	2	0	-4	3	1	3	-1	3	1	1	-3	1	1	2	-4	1	2	1	-4	2	0	0	-4	2	0	3	-3	1	3	3	-1
8000	2	1	1	-4	2	3	0	-3	1	3	1	-3	1	1	3	-3	3	0	3	-2	3	0	2	-3	3	2	0	-3	3	3	0	-2

FIG 18(5)

8032	1	0	2	-4	3	2	2	-2	2	3	2	-2	0	0	2	-4	2	2	3	-2	2	2	2	-3	2	1	0	-4	2	0	1	-4
8064	0	3	2	-3	0	3	3	-2	-3	3	1	-1	-1	3	3	-1	0	2	1	-4	0	1	2	-4	-3	1	3	-1	-1	2	0	-4
8096	0	2	3	-3	-1	3	1	-3	-1	1	3	-3	-3	1	1	-3	-1	2	1	-4	-2	3	2	-2	-1	1	2	-4	-2	2	3	-2
8128	-2	2	2	-3	-2	1	0	-4	-2	3	0	-3	-2	1	1	-4	-3	3	0	-2	-3	2	0	-3	-4	2	1	-1	-3	2	2	-2
8160	-4	1	2	-1	-4	1	1	-2	-4	2	0	-1	-4	1	0	-2	0	2	0	-4	2	-1	0	-4	0	-2	0	-4	2	-2	3	-2
8192	2	-2	2	-3	2	-3	2	-2	2	-4	0	-1	0	-4	0	-2	1	-2	0	-4	3	-2	0	-3	1	-4	0	-2	3	-3	0	-2
8224	0	-1	2	-4	3	-1	3	-1	2	-4	1	-1	0	-2	1	-4	2	-1	1	-4	0	-4	2	-1	1	-2	1	-4	0	-4	1	-2
8256	1	-4	2	-1	1	-4	1	-2	3	-1	1	-3	1	-1	2	-4	3	-3	1	-1	0	-2	3	-3	0	-3	3	-2	0	-3	2	-3
8288	1	-1	3	-3	2	-3	0	-3	1	-3	3	-1	1	-3	1	-3	3	-2	2	-2	-4	-1	0	-2	-2	-4	0	0	-4	-2	0	-1
8320	-3	-1	1	-3	-1	0	2	-4	-1	-2	0	-4	-4	0	0	-2	-1	-4	0	-2	-1	-1	2	-4	-1	-2	1	-4	-1	-4	2	-1
8352	-1	-4	1	-2	-1	-1	3	-3	-2	0	1	-4	-2	0	3	-3	-2	-1	0	-4	-2	-1	1	-4	-2	-3	0	-3	-2	-2	3	-2
8384	-2	-4	1	-1	-2	-2	2	-3	-1	-3	3	-1	-3	0	3	-2	-2	-3	2	-2	-3	0	2	-3	-3	-2	0	-3	-2	-4	0	-1
8416	-3	-3	0	-2	-1	-3	1	-3	-3	-2	2	-2	-4	-1	2	-1	-4	-1	1	-2	-3	-3	1	-1	-4	-2	1	-1	-3	-1	3	-1
8448	-4	0	2	-1	-4	0	1	-2	-2	0	0	-4	2	0	-4	-1	1	2	-4	0	3	3	-2	0	3	3	-1	-1	1	0	-2	-4
8480	3	2	-3	0	1	0	-4	-2	3	0	-2	-3	3	0	-3	-2	3	1	-1	-3	2	3	-3	0	2	1	-1	-4	2	1	-4	-1
8512	3	1	-3	-1	2	0	-3	-3	1	2	-1	-4	1	2	-4	-1	1	1	-2	-4	1	1	-4	-2	2	0	-4	0	1	3	-1	-3
8544	1	3	-3	-1	1	1	-3	-3	3	2	-2	-2	2	3	-2	-2	0	0	-2	-4	2	2	-2	-3	2	2	-3	-2	2	1	-4	0
8576	0	0	-4	-2	2	0	-1	-4	0	2	-1	-4	0	2	-4	-1	0	1	-2	-4	0	1	-4	-2	-3	3	-1	-1	-1	3	-1	-3
8608	-1	3	-3	-1	-1	2	-4	0	0	3	-2	-3	0	3	-3	-2	-4	2	0	0	0	2	-3	-3	-1	2	-1	-4	-1	2	-4	-1
8640	-1	1	-2	-4	-2	3	-2	-2	-1	1	-4	-2	-1	1	-3	-3	-2	2	-2	-3	-2	2	-3	-2	-2	1	-4	0	-2	3	-3	0
8672	-2	1	-1	-4	-2	1	-4	-1	-3	1	-1	-3	-3	3	-2	0	-3	2	-3	0	-3	1	-3	-1	-4	2	-1	-1	-3	2	-2	-2
8704	-4	1	-1	-2	-4	1	-2	-1	-4	2	-1	0	-4	1	-2	0	0	2	-4	0	2	-1	-4	0	0	-2	-4	0	2	-2	-2	-3
8736	2	-2	-3	-2	2	-3	-2	-2	2	-4	-1	0	0	-4	-2	0	1	-2	-4	0	3	-2	-3	0	1	-4	-2	0	3	-3	-2	0
8768	2	-1	-1	-4	2	-1	-4	-1	0	-1	-2	-4	2	-3	-3	0	0	-1	-4	-2	2	-4	-1	-1	0	-2	-1	-4	0	-2	-4	-1
8800	1	-1	-2	-4	1	-1	-4	-2	3	-1	-1	-3	1	-2	-1	-4	1	-2	-4	-1	0	-4	-1	-2	3	-1	-3	-1	1	-4	-1	-2
8832	0	-4	-2	-1	1	-4	-2	-1	3	-3	-1	-1	0	-2	-3	-3	0	-3	-2	-3	0	-3	-3	-2	1	-1	-3	-3	1	-3	-1	-3
8864	1	-3	-3	-1	3	-2	-2	-2	-4	-1	-2	0	-4	-2	-1	0	-3	-3	-1	-1	-3	-1	-1	-3	-1	0	-2	-4	-1	0	-4	-2
8896	-1	-2	-4	0	-1	-4	-2	0	-1	-1	-2	-4	-1	-1	-4	-2	-1	-2	-1	-4	-1	-2	-4	-1	-1	-4	-1	-2	-1	-4	-2	-1
8928	-3	-1	-3	-1	-2	0	-1	-4	-2	0	-4	-1	-2	0	-3	-3	-2	-1	-1	-4	-2	-1	-4	0	-2	-1	-4	-1	-1	-1	-3	-3
8960	-2	-3	-3	0	-2	-4	-1	-1	-2	-2	-2	-3	-2	-2	-3	-2	-3	0	-2	-3	-2	-3	-2	-2	-3	0	-3	-2	-3	-2	-3	0
8992	-2	-4	-1	0	-3	-3	-2	0	-3	-2	-2	-2	-1	-3	-1	-3	-4	-1	-1	-2	-4	-1	-2	-1	-4	-2	-1	-1	-1	-3	-3	-1
9024	-4	0	-1	-2	-4	0	-2	-1	-2	0	-4	0	-4	0	-2	0	-4	-2	0	0	3	3	2	1	1	3	2	3	3	3	1	2
9056	2	3	3	1	2	3	1	3	3	2	2	3	1	2	3	3	3	1	2	3	2	3	2	3	1	3	3	2	2	1	3	3
9088	2	3	3	2	3	1	3	2	3	2	1	3	3	3	2	2	3	2	3	2	2	2	3	3	3	2	3	1	-2	1	3	3
9120	-3	3	2	1	-1	3	2	3	-3	2	3	1	-3	2	1	3	-4	2	1	2	-4	3	0	0	-2	3	2	3	-4	3	1	0
9152	-1	2	3	3	-2	3	1	3	-1	3	3	2	-3	2	2	3	-3	2	3	2	-2	2	3	3	-2	3	3	1	-3	1	3	2
9184	-4	1	2	2	-2	3	3	2	-3	3	1	2	-3	1	2	3	-4	1	3	0	-3	3	2	2	-4	2	2	1	-4	2	2	0
9216	3	-4	1	0	3	-3	2	1	3	-3	2	2	3	-2	2	3	3	-4	0	0	2	-4	2	0	2	-4	2	1	3	-1	2	3
9248	1	-3	3	2	2	-3	3	1	0	-4	1	3	2	-2	3	3	1	-4	2	2	2	-1	3	3	2	-3	2	3	0	-4	3	1
9280	1	-3	2	3	1	-2	3	3	3	-1	3	2	3	-2	3	2	3	-2	1	3	2	-3	1	3	0	-4	3	0	3	-2	3	1
9312	2	-4	1	2	3	-3	1	2	2	-3	3	2	1	-4	3	0	0	-4	2	2	-2	-4	2	1	-4	0	1	3	-1	-4	3	0
9344	-3	-3	1	2	-3	-3	2	2	-2	-3	3	2	-4	-2	1	2	-2	-4	1	2	-3	-2	1	3	-4	-1	2	2	-3	-3	2	1
9376	-4	-2	2	0	-3	-2	2	3	-3	-2	3	1	-4	-3	1	0	-3	-4	1	0	-3	-1	3	2	-2	-4	2	0	-3	-2	3	2
9408	-3	-1	2	3	-1	-3	3	2	-2	-3	1	3	-1	-2	3	3	-4	-1	3	0	-2	-2	3	3	-4	0	3	1	-4	-2	2	1
9440	-2	-3	3	1	-2	-1	3	3	-1	-4	2	2	-4	-3	0	0	-4	0	3	0	-2	-3	2	3	-4	0	2	2	-1	-3	2	3
9472	3	1	-2	3	3	2	-3	2	1	3	-3	2	1	3	-2	3	1	2	-4	2	1	2	-3	3	3	2	-1	3	2	1	-4	2
9504	3	2	-3	1	2	2	-3	3	0	0	-4	3	3	3	-1	2	3	0	-4	1	2	3	-1	3	2	2	-4	1	2	1	-3	3
9536	1	0	-4	3	3	1	-3	2	3	3	-2	1	3	3	-2	2	2	3	-3	1	2	0	-4	2	2	3	-2	3	2	3	-3	2
9568	3	2	-2	3	0	1	-4	3	-2	2	-3	3	-4	1	-2	2	-3	1	-3	2	-3	3	-2	2	-3	2	-3	2	-4	2	0	2
9600	-4	2	-1	2	-4	2	-2	1	-4	3	0	1	-1	3	-2	3	-1	3	-3	2	-3	3	-1	2	-2	3	-2	3	-1	2	-3	3
9632	-3	3	-2	1	-2	3	-3	1	-2	1	-4	2	-3	1	-2	3	-3	2	-2	3	-4	1	0	3	-1	2	-4	2	0	2	-4	2

FIG 18(6)

9664	-2	2	-4	1	-3	2	-1	3	-2	3	-1	3	-2	3	-3	2	-2	1	-3	3	-3	2	-3	1	0	3	-4	1	2	-4	0	2
9696	2	-3	-2	3	0	-3	-4	1	1	-4	0	3	1	-2	-3	3	0	-1	-4	3	0	-4	0	3	2	-1	-4	2	0	-4	-3	1
9728	0	-4	-1	3	0	-4	-2	2	1	-3	-2	3	1	-3	-3	2	3	-2	-2	3	2	-2	-4	1	2	-4	-1	2	0	-2	-4	2
9760	3	-1	-2	3	1	-2	-4	2	3	-1	-3	2	2	-3	-3	1	1	-4	-2	2	2	-4	-2	1	3	-2	-1	3	3	-2	-3	1
9792	2	-3	-1	3	3	-3	-1	2	3	-3	-2	1	2	-3	-3	2	2	-2	-3	3	3	-2	-3	2	3	-3	-2	2	2	-1	-3	3
9824	3	-4	0	1	-4	-2	0	2	-2	0	-4	2	-2	-3	-2	3	-2	-1	-3	3	-1	0	-4	3	-2	-2	-3	3	-2	-3	-3	2
9856	-2	-3	-1	3	-2	-3	-3	1	-1	-4	0	3	-3	-2	-2	3	-3	-2	-3	2	-3	-3	-2	2	-3	-2	-1	3	-3	-4	0	1
9888	-3	-3	-2	1	-4	0	-1	3	-4	0	-3	1	-4	-3	0	1	-3	0	-4	1	-4	-2	-2	1	-4	-1	0	3	-1	-2	-3	3
9920	-3	-1	-2	3	-3	-2	-3	1	-4	0	-2	2	-4	-2	-1	2	-1	-3	-2	3	-1	-2	-4	2	-1	-4	-2	2	-1	-3	-3	2
9952	-3	-1	-3	2	-3	-4	0	0	-2	-1	-4	2	-2	-4	-2	1	-2	-2	-4	1	-3	-3	-1	2	-2	-4	-1	2	-2	-4	0	2
9984	-4	-1	-2	2	-4	0	0	3	2	2	3	-3	1	3	2	-3	1	2	3	-3	2	0	2	-4	1	3	3	-2	3	3	1	-2
10016	3	2	3	-1	3	0	1	-4	3	2	1	-3	3	0	0	-4	2	2	1	-4	2	1	2	-4	3	1	3	-2	3	1	2	-3
10048	1	2	2	-4	1	0	3	-4	1	3	0	-4	0	0	3	-4	3	1	0	-4	2	3	3	-1	2	3	1	-3	3	3	2	-2
10080	3	2	3	-2	3	2	2	-3	3	3	2	-1	2	1	3	-3	2	2	0	-4	2	3	3	-2	2	3	2	-3	-3	1	3	-2
10112	-2	1	3	-3	0	3	1	-4	0	1	3	-4	-3	1	2	-3	-1	3	0	-4	-2	3	3	-2	-2	3	2	-3	-2	2	3	-3
10144	-2	2	0	-4	-3	3	2	-2	-3	2	3	-2	-3	2	2	-3	-3	1	0	-4	-4	2	0	-2	-4	3	0	-1	-1	3	3	-2
10176	-4	1	0	-3	-3	3	2	-1	-1	3	2	-3	-3	3	1	-2	-1	2	3	-3	0	3	0	-4	-3	2	3	-1	-1	2	2	-4
10208	-2	2	1	-4	-3	2	1	-3	-2	1	2	-4	-2	3	3	-1	-2	3	1	-3	-4	2	2	-1	-4	2	1	-2	0	2	2	-4
10240	-4	1	2	-2	2	-3	3	-1	2	-2	3	-3	2	-3	1	-3	2	-3	3	-2	2	-3	2	-3	1	-3	0	-4	1	-4	0	-3
10272	0	-1	3	-4	2	-2	0	-4	0	-3	1	-4	0	-4	3	-1	0	-4	1	-3	2	-4	0	-2	0	-2	2	-4	0	-4	2	-2
10304	1	-2	3	-3	1	-3	3	-2	1	-3	2	-3	3	-1	0	-4	2	-1	2	-4	2	-2	1	-4	2	-4	2	-1	2	-4	1	-2
10336	3	-1	3	-2	3	-1	2	-3	1	-2	2	-4	1	-4	2	-2	3	-2	3	-1	0	-3	0	-4	3	-2	1	-3	0	-4	0	-3
10368	3	-3	2	-1	3	-3	1	-2	3	-2	3	-2	3	-2	2	-3	2	-1	3	-3	3	-3	2	-2	3	-4	0	-1	-4	-1	2	-2
10400	-4	-2	2	-1	-3	-2	3	-1	-4	0	0	-3	-1	0	3	-4	-2	-1	3	-3	-1	-3	0	-4	-3	-3	2	-1	-1	-4	0	-3
10432	-2	-3	3	-1	-2	-2	3	-3	-2	0	2	-4	-2	-3	3	-2	-2	-3	2	-3	-2	-3	1	-3	-3	-2	1	-3	-2	-2	0	-4
10464	-2	-4	0	-2	-3	0	1	-4	-3	-1	0	-4	-3	-2	3	-2	-4	0	2	-2	-3	-2	2	-3	-3	-3	2	-2	-4	-2	0	-2
10496	-3	-4	0	-1	-4	0	3	-1	-4	0	1	-3	-3	-1	3	-2	-4	-1	0	-3	-4	-3	0	-1	-3	-1	2	-3	-1	-2	3	-3
10528	-3	-3	1	-2	-1	-3	3	-2	-1	-3	2	-3	-1	-2	2	-4	-1	-4	2	-2	-2	-1	2	-4	-2	-2	1	-4	-2	-4	2	-1
10560	-2	-4	1	-2	-3	0	0	-4	-4	-2	1	-2	2	3	-2	-3	2	3	-3	-2	2	2	-3	-3	1	3	-4	0	1	0	-3	-4
10592	1	0	-4	-3	2	0	-2	-4	2	0	-4	-2	1	3	-2	-3	1	3	-3	-2	2	2	-4	0	1	2	-3	-3	3	3	-1	-2
10624	3	3	-2	-1	3	0	-4	0	3	2	-1	-3	2	2	-1	-4	2	2	-4	-1	3	2	-3	-1	2	1	-2	-4	2	1	-4	-2
10656	3	1	-2	-3	3	1	-3	-2	1	2	-2	-4	1	2	-4	-2	0	0	-3	-4	0	0	-4	-3	2	3	-1	-3	3	3	-2	-2
10688	2	3	-3	-1	3	2	-2	-3	3	2	-3	-2	3	1	-4	0	3	0	-1	-4	3	0	-4	-1	2	1	-3	-3	-2	1	-3	-3
10720	0	3	-1	-4	0	3	-4	-1	0	1	-3	-4	0	1	-4	-3	-1	3	-4	0	0	2	-2	-4	0	2	-4	-2	-3	1	-2	-3
10752	-2	3	-2	-3	-2	3	-3	-2	-2	2	-3	-3	-2	2	-4	0	-3	3	-2	-2	-3	2	-2	-3	-3	2	-3	-2	-3	1	-4	0
10784	-4	2	-2	0	-3	1	-3	-2	-4	3	-1	0	-4	1	-3	0	-1	3	-2	-3	-1	3	-3	-2	-3	3	-1	-2	-1	2	-3	-3
10816	-3	3	-2	-1	0	3	-4	0	-1	2	-2	-4	-1	2	-4	-2	-2	2	-1	-4	-2	2	-4	-1	-2	1	-2	-4	-2	1	-4	-2
10848	-3	2	-1	-3	-2	3	-1	-3	-4	2	-1	-2	-3	2	-3	-1	-4	2	-2	-1	-2	3	-3	-1	-4	1	-2	-2	2	-2	-3	-3
10880	2	-3	-2	-3	2	-3	-3	-2	2	-3	-1	-3	1	-3	-4	0	2	-3	-3	-1	1	-4	-3	0	0	-1	-3	-4	0	-1	-4	-3
10912	0	-3	-1	-4	2	-2	-4	0	0	-3	-4	-1	0	-4	-1	-3	2	-4	-2	0	0	-4	-3	-1	0	-2	-2	-4	0	-2	-4	-2
10944	0	-4	-2	-2	1	-2	-3	-3	1	-3	-2	-3	1	-3	-3	-2	2	-1	-2	-4	2	-1	-4	-2	2	-2	-1	-4	2	-2	-4	-1
10976	2	-4	-1	-2	2	-4	-2	-1	1	-2	-2	-4	1	-2	-4	-2	3	-1	-2	-3	1	-4	-2	-2	3	-1	-3	-2	0	-3	-4	0
11008	0	-4	-3	0	3	-2	-1	-3	3	-2	-3	-1	3	-3	-1	-2	3	-3	-2	-1	3	-1	-4	0	3	-2	-2	-3	3	-2	-3	-2
11040	3	-3	-2	-2	3	-4	-1	0	2	-1	-3	-3	-4	0	-3	0	-4	-1	-2	-2	-4	-2	-1	-2	-4	-2	-2	-1	-2	-1	-3	-3
11072	-1	0	-3	-4	-1	0	-4	-3	-1	-3	-4	0	-1	-4	-3	0	-2	-2	-3	-3	-2	0	-2	-4	-2	-3	-2	-3	-2	0	-4	-2
11104	-2	-3	-3	-2	-2	-2	-4	0	-2	-4	-2	0	-2	-3	-1	-3	-3	0	-1	-4	-3	0	-4	-1	-3	-1	-4	0	-3	-2	-2	-3
11136	-4	0	-2	-2	-3	-2	-3	-2	-2	-3	-3	-1	-3	-3	-2	-2	-4	-2	-2	0	-3	-4	-1	0	-3	-3	-1	-2	-4	0	-1	-3
11168	-4	0	-3	-1	-4	-1	-3	0	-4	-3	-1	0	-3	-2	-1	-3	-1	-2	-3	-3	-1	-2	-2	-4	-3	-1	-2	-3	-1	-2	-4	-2
11200	-1	-3	-2	-3	-1	-4	-2	-2	-1	-3	-3	-2	-2	-1	-2	-4	-3	-2	-3	-1	-2	-1	-4	-2	-2	-2	-1	-4	-2	-2	-4	-1
11232	-2	-4	-1	-2	-2	-4	-2	-1	-3	0	-4	0	-3	-1	-3	-2	-3	-3	-2	-1	3	3	2	3	3	1	3	3	3	0	3	3
11264	3	3	1	3	3	3	3	0	1	3	3	3	2	3	3	3	3	3	3	2	3	3	3	1	3	2	3	3	-4	1	3	1

FIG 18(7)

11296	0	3	3	3	-2	3	3	3	-3	3	2	3	-4	3	2	1	-4	3	1	2	-4	2	1	3	-1	3	3	3	-4	1	3	2
11328	-3	3	1	3	-4	3	2	0	-3	3	3	1	-4	2	3	0	-3	3	3	2	-4	3	1	1	-3	3	3	0	-4	1	2	3
11360	-3	2	3	3	-4	2	2	2	-4	2	3	1	-4	1	1	3	-3	1	3	3	1	-4	3	1	1	-4	1	3	3	-4	2	1
11392	2	-4	1	3	0	-3	3	3	3	-4	1	1	3	-4	2	0	3	-1	3	3	2	-3	3	3	0	-4	3	2	1	-3	3	3
11424	1	-4	3	2	3	-3	2	3	3	-3	3	2	0	-4	2	3	3	-3	3	1	3	-3	3	0	2	-4	3	0	3	-2	3	3
11456	1	-4	2	3	2	-4	3	1	2	-4	2	2	3	-4	1	2	3	-3	1	3	-2	-4	2	2	-4	0	2	3	-3	-3	2	3
11488	-4	-1	2	3	-2	-4	1	3	-4	-2	3	1	-4	-2	3	0	-1	-4	2	3	-4	-1	3	1	-4	-3	1	2	-3	-4	2	0
11520	-2	-4	3	0	-3	-3	3	0	-4	-2	1	3	-3	-3	3	2	-3	-4	1	2	-1	-4	1	3	-1	-3	3	3	-3	-3	3	1
11552	-3	-4	2	1	-3	0	3	3	-4	-1	1	3	-4	-1	3	2	-4	-3	1	1	-3	-3	1	3	-4	-4	0	0	-4	-3	2	1
11584	-3	-1	3	3	-4	-3	2	0	-4	0	3	2	-2	-4	3	1	-4	-2	2	2	-3	-2	3	3	-3	-4	1	1	-1	-4	3	1
11616	-2	-3	3	3	-1	-4	3	2	3	3	-3	2	3	3	0	3	1	1	-4	3	2	2	-4	2	3	3	-3	1	3	1	-3	3
11648	3	1	-4	1	2	1	-4	3	1	3	-4	2	3	0	-3	3	3	3	-1	3	3	0	-4	2	1	3	-4	1	3	2	-3	3
11680	1	2	-4	3	1	3	-3	3	2	3	-4	1	2	3	-3	3	3	3	-2	3	3	1	-4	2	2	0	-4	3	3	2	-4	1
11712	-4	3	0	2	-2	3	-3	3	-3	3	-2	3	-4	3	-2	1	-4	2	0	3	-2	2	-4	2	-3	3	-3	2	-3	3	-3	1
11744	-4	1	-2	3	-1	2	-4	3	-3	2	-4	1	-3	1	-4	2	0	2	-4	3	-1	3	-3	3	-4	2	-1	3	-4	2	-3	1
11776	-4	3	-1	1	-4	3	-1	2	-3	1	-4	1	-1	3	-4	2	-1	3	-4	1	-2	1	-4	3	-4	1	-3	2	-2	3	-4	1
11808	-3	3	-1	3	0	3	-3	3	-3	1	-3	3	-4	1	-3	1	-4	1	-1	3	0	3	-4	2	-3	3	0	3	-1	1	-4	3
11840	-3	2	-3	3	-4	2	-2	2	2	-1	-4	3	2	-3	-4	1	2	-4	-3	1	3	-4	-1	2	3	-1	-4	1	1	-3	-4	2
11872	3	-3	0	3	3	-4	-1	1	3	-4	0	2	3	-3	-1	3	2	-3	-3	3	3	-3	-3	2	3	-4	-2	1	2	-2	-4	2
11904	3	-3	-3	1	3	-1	-4	2	0	-4	-2	3	1	-1	-4	3	1	-4	-3	2	1	-4	-2	3	1	-3	-4	1	1	-3	-3	3
11936	3	-2	-4	1	2	-4	-1	3	1	-4	-1	3	0	-3	-4	2	1	-4	-3	1	1	-2	-4	3	2	-4	0	3	0	-3	-3	3
11968	3	-1	-3	3	3	-2	-3	3	0	-2	-4	3	0	-4	-3	2	2	-4	-2	2	3	-3	-2	3	-4	-1	-3	1	-1	-1	-4	3
12000	-4	0	-2	3	-1	-3	-4	1	-4	-2	0	3	-4	-3	0	2	-1	-2	-4	3	-3	0	-3	3	-1	-3	-4	2	-1	-4	-2	3
12032	-1	-4	-3	1	-3	-4	-1	1	-2	-1	-4	3	-4	-2	-2	2	-2	-3	-4	1	-2	-4	-1	3	-2	-4	-3	1	-3	-2	-4	1
12064	-3	-3	0	3	-3	-4	-1	2	-2	-4	0	3	-3	-4	0	2	-1	-3	-3	3	-4	-1	-2	3	-4	-1	-3	2	-2	-2	-4	2
12096	-4	-2	-1	3	-4	-2	-3	1	-2	-4	-2	2	-4	-3	-1	2	-4	-3	-2	1	-1	-4	-3	2	-3	-4	-2	1	-3	-1	-3	3
12128	-1	-4	-1	3	-3	-3	-2	3	-4	-1	-1	3	-3	-3	-1	3	-3	-3	-3	1	-3	-1	-4	1	-2	-3	-3	3	-2	0	-4	3
12160	-4	0	-3	2	-3	-2	-3	3	-3	0	-4	2	-3	-3	-3	2	-4	-3	-1	1	-3	-1	-4	2	3	3	3	-2	3	0	3	-3
12192	3	1	1	-4	2	3	1	-4	2	1	3	-4	1	3	2	-4	3	1	3	-3	1	3	1	-4	2	2	2	-4	1	1	3	-4
12224	3	0	2	-4	1	3	3	-3	3	3	1	-3	3	3	3	-1	3	2	0	-4	3	1	2	-4	2	3	0	-4	2	0	3	-4
12256	3	3	0	-3	1	2	3	-4	3	3	2	-3	2	3	3	-3	3	2	3	-3	3	2	1	-4	-4	2	0	-3	-3	3	2	-3
12288	-4	3	0	-2	-1	3	2	-4	-1	2	3	-4	-4	2	2	-2	-2	3	1	-4	-2	1	3	-4	-2	3	3	-3	-3	2	1	-4
12320	-3	1	2	-4	-4	2	3	-1	-4	3	1	-1	-4	3	2	-1	-4	3	1	-2	-1	3	3	-3	-4	2	1	-3	-3	1	1	-4
12352	-4	1	3	-2	-4	1	2	-3	-2	2	2	-4	-3	3	3	-1	0	3	3	-3	-3	3	1	-3	-4	1	3	-1	-3	3	0	-3
12384	-3	1	3	-3	0	3	2	-4	0	2	3	-4	-4	1	1	-3	-1	1	3	-4	-2	3	0	-4	-3	3	3	-2	-1	3	1	-4
12416	-3	2	3	-3	-3	2	0	-4	3	-1	2	-4	3	-4	2	-1	3	-4	1	-2	2	-3	1	-4	2	-1	3	-4	2	-4	3	-1
12448	2	-4	1	-3	3	-1	1	-4	1	-2	3	-4	1	-3	2	-4	1	-4	3	-2	1	-4	2	-3	3	-3	0	-3	3	-4	1	-1
12480	3	-1	3	-3	3	-3	3	-1	3	-3	1	-3	2	-2	2	-4	1	-1	3	-4	2	-4	2	-2	3	-2	1	-4	1	-3	1	-4
12512	1	-3	3	-3	1	-4	3	-1	3	-3	3	-2	1	-4	1	-3	3	-2	0	-4	3	-4	0	-2	2	-3	0	-4	0	-3	3	-3
12544	2	-4	0	-3	0	-3	2	-4	0	-2	3	-4	3	-2	3	-3	0	-4	3	-2	3	-3	2	-3	0	-4	2	-3	2	-3	3	-3
12576	0	-4	0	-4	-3	-2	0	-4	-3	-3	0	-3	-1	-3	1	-4	-4	0	3	-2	-4	0	2	-3	-4	-2	0	-3	-4	-3	0	-2
12608	-1	-4	3	-1	-3	-4	1	-1	-1	-2	3	-4	-1	-3	2	-4	-1	-4	1	-3	-1	-4	3	-2	-1	-4	2	-3	-2	-1	3	-4
12640	-2	-3	1	-4	-2	-4	3	-1	-2	-4	1	-3	-4	-3	1	-1	-3	-1	2	-4	-3	-2	1	-4	-3	-4	0	-2	-3	-4	2	-1
12672	-3	-4	1	-2	-4	-1	3	-1	-1	-3	3	-3	-4	-1	3	-2	-4	-1	2	-3	-2	-2	2	-4	-4	-2	3	-1	-4	-2	1	-3
12704	-4	-3	2	-1	-2	-4	2	-2	-4	-3	1	-2	-3	0	3	-3	-3	-1	3	-3	-3	-1	1	-4	-4	0	0	-4	-4	-1	1	-3
12736	-3	-3	3	-1	-3	-3	1	-3	-2	0	3	-4	-2	-3	3	-3	-2	-3	0	-4	-2	-4	0	-3	-4	-2	2	-2	-3	-2	3	-3
12768	-3	-3	3	-2	-3	-3	2	-3	-3	0	2	-4	-1	-1	3	-4	3	2	-1	-4	3	2	-4	-1	3	1	-2	-4	3	1	-4	-2
12800	3	3	-3	0	3	1	-1	-4	2	3	-1	-4	2	3	-4	-1	3	1	-4	-1	2	1	-3	-4	2	1	-4	-3	3	0	-3	-3
12832	1	3	-2	-4	1	3	-4	-2	1	2	-3	-4	1	2	-4	-3	3	3	-3	-1	3	1	-3	-3	1	3	-1	-4	1	3	-4	-1
12864	2	2	-2	-4	1	1	-3	-4	2	2	-4	-2	1	1	-4	-3	1	3	-3	-3	3	3	-1	-3	3	2	-4	0	3	0	-2	-4
12896	3	0	-4	-2	2	3	-4	0	2	0	-3	-4	2	0	-4	-3	0	0	-4	-4	3	3	-2	-3	3	3	-3	-2	3	2	-3	-3

FIG 18(8)

12928	2	3	-3	-3	-4	3	-2	0	-4	1	-1	-3	-4	2	-3	0	-1	3	-2	-4	-1	3	-4	-2	-1	2	-3	-4	-1	2	-4	-3
12960	-4	1	-3	-1	-2	3	-1	-4	-2	3	-4	-1	-2	1	-3	-4	-2	1	-4	-3	-3	2	-1	-4	-3	2	-4	-1	-3	1	-2	-4
12992	-3	1	-4	-2	-4	3	-1	-2	-4	3	-2	-1	-3	3	-3	0	-4	2	-1	-3	-1	3	-3	-3	-4	2	-3	-1	-4	1	-2	-3
13024	-4	1	-3	-2	-3	1	-1	-4	-2	2	-2	-4	-2	2	-4	-2	-4	3	-1	-1	-3	1	-4	-1	-3	3	-1	-3	0	3	-3	-3
13056	-3	3	-3	-1	-3	1	-3	-3	0	3	-2	-4	0	3	-4	-2	0	2	-3	-4	0	2	-4	-3	-1	3	-1	-4	-1	3	-4	-1
13088	-2	3	-4	0	-2	3	-3	-3	-1	1	-3	-4	-4	2	-2	-2	-3	3	-2	-3	-3	3	-3	-2	-3	2	-3	-3	-1	1	-4	-3
13120	-3	2	-4	0	3	-1	-2	-4	3	-1	-4	-2	3	-2	-1	-4	3	-2	-4	-1	3	-4	-1	-2	3	-4	-2	-1	2	-1	-3	-4
13152	2	-1	-4	-3	2	-3	-1	-4	2	-3	-4	-1	2	-4	-1	-3	2	-4	-3	-1	3	-1	-1	-4	1	-2	-3	-4	1	-2	-4	-3
13184	3	-1	-4	-1	1	-3	-2	-4	1	-3	-4	-2	1	-4	-2	-3	1	-4	-3	-2	3	-3	-3	0	3	-4	-1	-1	3	-1	-3	-3
13216	3	-3	-1	-3	3	-3	-3	-1	2	-2	-2	-4	2	-2	-4	-2	1	-1	-3	-4	2	-4	-2	-2	1	-1	-4	-3	1	-3	-1	-4
13248	1	-3	-3	-3	1	-3	-4	-1	1	-4	-1	-3	1	-4	-3	-1	3	-2	-4	0	3	-4	-2	0	2	-3	-4	0	2	-4	-3	0
13280	0	-3	-3	-3	0	-4	-4	0	0	-2	-3	-4	0	-2	-4	-3	0	-3	-2	-4	0	-3	-4	-2	3	-2	-3	-3	3	-3	-2	-3
13312	3	-3	-3	-2	0	-4	-2	-3	0	-4	-3	-2	2	-3	-3	-3	-3	0	-4	-2	-3	-2	-4	0	-1	-1	-3	-4	-3	-4	-2	0
13344	-1	-1	-4	-3	-1	-3	-1	-4	-4	0	-2	-3	-4	0	-3	-2	-3	-3	-3	0	-4	-2	-3	0	-1	-3	-4	-1	-4	-3	-2	0
13376	-4	-1	-1	-3	-1	-2	-3	-4	-1	-2	-4	-3	-1	-3	-2	-4	-1	-3	-4	-2	-1	-4	-2	-3	-1	-4	-3	-2	-1	-4	-1	-3
13408	-2	-1	-3	-4	-2	-1	-4	-3	-2	-3	-1	-4	-2	-3	-4	-1	-3	-4	-1	-1	-2	-4	-1	-3	-2	-4	-3	-1	-1	-4	-3	-1
13440	-3	-1	-2	-4	-3	-1	-4	-2	-3	-2	-1	-4	-3	-2	-4	-1	-3	-4	-1	-2	-3	-4	-2	-1	-1	-3	-3	-3	-4	-1	-2	-3
13472	-4	-1	-3	-2	-2	-2	-2	-4	-4	-2	-1	-3	-2	-2	-4	-2	-4	-2	-3	-1	-4	-3	-1	-2	-2	-4	-2	-2	-4	-3	-2	-1
13504	-4	-1	-3	-1	-3	0	-3	-3	-3	-1	-3	-3	-4	0	-4	0	-3	-1	-1	-4	-3	-3	-1	-3	-3	-3	-3	-1	-2	0	-3	-4
13536	-2	-3	-3	-3	-2	0	-4	-3	-2	-3	-4	0	-3	-1	-4	-1	-2	-4	-3	0	-3	-2	-3	-3	-4	-2	-2	-2	-3	-3	-2	-3
13568	-3	0	-2	-4	-3	-3	-3	-2	-4	-3	-1	-1	3	3	3	3	-4	3	3	0	-4	3	1	3	-4	2	3	3	-4	3	2	3
13600	-4	3	3	2	-4	3	2	2	-4	3	3	1	-4	2	2	3	-3	3	3	3	-4	2	3	2	-4	1	3	3	3	-3	3	3
13632	3	-4	2	3	2	-4	3	3	3	-4	3	1	3	-4	1	3	3	-4	2	2	2	-4	3	2	0	-4	3	3	2	-4	2	3
13664	1	-4	3	3	3	-4	3	0	3	-4	3	2	-3	-4	3	0	-4	-4	1	1	-2	-4	2	3	-2	-4	3	2	-3	-4	1	3
13696	-4	-1	3	3	-3	-4	3	2	-3	-3	3	3	-4	-4	2	0	-4	-2	2	3	-4	-2	3	2	-3	-4	2	2	-2	-4	3	3
13728	-4	-3	3	2	-4	-4	1	0	-4	-2	3	3	-4	-3	3	0	-4	-4	2	1	-4	0	3	3	-4	-3	1	3	-1	-4	3	3
13760	-4	-3	2	3	-4	-3	3	1	-4	-4	1	2	-3	-4	2	3	-4	-3	2	2	-3	-4	3	1	3	1	-4	3	2	2	-4	3
13792	3	2	-4	2	3	3	-3	3	1	3	-4	3	3	3	-4	1	3	3	-4	2	3	2	-4	3	2	3	-4	2	3	0	-4	3
13824	2	3	-4	3	-4	1	-4	2	-3	2	-4	3	-4	2	-2	3	-4	2	-3	3	-2	3	-4	2	-3	3	-4	2	-3	3	-3	3
13856	-3	2	-4	2	-2	3	-4	3	-4	3	-3	1	-4	3	-1	3	-2	2	-4	3	-4	1	-3	3	-4	2	-3	2	-3	3	-4	1
13888	-4	1	-4	1	-4	3	-2	2	-3	1	-4	3	-4	3	0	3	-4	3	-3	2	-4	2	-4	1	-4	3	-2	3	-1	3	-4	3
13920	0	3	-4	3	0	-4	-4	1	3	-3	-4	2	2	-4	-3	2	2	-2	-4	3	0	-4	-4	2	3	-2	-4	2	3	-4	-2	2
13952	3	-4	-3	2	2	-4	-3	3	3	-3	-3	3	1	-4	-4	1	2	-3	-4	2	1	-4	-3	3	2	-4	-4	1	0	-3	-4	3
13984	0	-4	-3	3	2	-3	-4	3	1	-4	-4	2	3	-4	-2	3	3	-4	-1	3	3	-3	-4	1	3	-4	-3	1	3	-2	-4	3
14016	3	-4	0	3	3	-1	-4	3	2	-4	-2	3	1	-3	-4	3	-2	-4	-3	2	-4	-4	-1	1	-1	-4	-3	3	-2	-4	-3	3
14048	-2	-4	-4	1	-3	-2	-4	3	-3	-3	-4	2	-1	-4	-4	1	-3	-4	-3	1	-2	-4	-2	3	-3	-4	-2	3	-3	-4	-3	2
14080	-4	-3	-1	3	-4	-3	-2	2	-1	-3	-4	3	-3	-4	0	3	-4	-1	-3	3	-4	-3	-3	2	-4	-4	-1	2	-1	-4	-4	2
14112	-4	-2	-3	2	-4	0	-4	2	-4	-3	0	3	-3	-2	-4	2	-2	-3	-4	2	-4	-2	-4	1	-4	-3	-2	3	-4	-2	-3	3
14144	-4	-4	-2	1	-2	-2	-4	3	-4	0	-4	1	-4	-2	-2	3	-4	-3	-3	1	-3	-4	-2	2	-3	0	-4	3	-3	-3	-4	1
14176	-3	-4	-1	3	-4	-4	0	2	-4	-4	0	1	-3	-3	-3	3	-4	-1	-4	2	-2	-3	-4	3	-4	0	-3	3	-4	-1	-4	1
14208	-3	-1	-4	3	2	3	3	-4	3	3	1	-4	3	2	2	-4	2	2	3	-4	3	3	0	-4	3	3	3	-3	2	3	2	-4
14240	3	2	3	-4	3	1	3	-4	3	0	3	-4	3	3	2	-4	1	3	3	-4	-1	3	3	-4	-4	2	3	-2	-3	3	1	-4
14272	-3	1	3	-4	-4	2	2	-3	-4	3	3	-1	-3	2	3	-4	-3	3	2	-4	-4	1	3	-3	-4	3	1	-3	-4	3	3	-2
14304	-4	3	2	-3	-4	2	3	-3	-4	1	0	-4	-2	3	3	-4	-4	2	1	-4	-4	1	2	-4	-3	3	0	-4	-3	3	3	-3
14336	-2	3	2	-4	-4	3	0	-3	-4	1	1	-4	-4	3	2	-2	0	3	3	-4	-2	2	3	-4	-4	2	0	-4	-3	2	2	-4
14368	1	-4	3	-3	1	-4	1	-4	2	-4	3	-3	3	-4	3	-2	3	-4	2	-3	3	-4	2	-2	2	-2	3	-4	2	-3	2	-4
14400	2	-3	3	-4	3	-3	0	-4	2	-4	1	-4	1	-4	2	-4	3	-4	0	-3	2	-4	3	-2	2	-4	2	-3	0	-3	3	-4
14432	1	-4	0	-4	0	-4	3	-3	3	-3	3	-3	2	-4	0	-4	0	-4	1	-4	0	-4	2	-4	3	-2	3	-4	3	-3	2	-4
14464	3	-2	2	-4	3	-1	3	-4	3	-3	1	-4	3	-4	3	-1	3	-4	1	-3	1	-3	3	-4	-4	-3	0	-3	-1	-3	3	-4
14496	-4	-4	1	-1	-3	-4	3	-2	-1	-4	3	-3	-2	-3	2	-4	-4	-2	3	-2	-3	-1	3	-4	-4	-3	2	-2	-2	-4	3	-2
14528	-2	-3	3	-4	-4	-2	2	-3	-2	-4	3	-3	-2	-4	2	-3	-3	-2	3	-4	-3	-3	2	-4	-4	-1	3	-3	-3	-4	2	-3

FIG 18(9)

14560	-4	-3	3	-1	-4	-3	1	-3	-4	-2	3	-3	-4	-3	3	-2	-4	-3	2	-3	-1	-4	2	-4	-2	-4	1	-4	-3	-2	2	-4
14592	-4	0	1	-4	-4	-1	2	-4	-3	-4	1	-3	-4	-2	1	-4	-4	-4	2	-1	-4	-4	1	-2	-1	-4	1	-4	-3	-4	2	-2
14624	-3	-4	3	-1	-2	-4	0	-4	-3	0	3	-4	-3	-3	1	-4	-3	-3	3	-3	-3	-3	0	-4	-4	0	2	-4	-1	-4	0	-4
14656	-4	-2	0	-4	-3	-4	0	-3	-4	-4	0	-2	-4	-1	0	-4	-4	0	3	-3	-4	-1	1	-4	-2	-2	3	-4	-4	-4	0	-1
14688	3	2	-2	-4	3	2	-4	-2	3	3	-4	-2	3	2	-3	-4	3	2	-4	-3	2	3	-3	-4	2	3	-4	-3	2	3	-2	-4
14720	2	3	-4	-2	2	2	-3	-4	2	2	-4	-3	3	3	-4	0	2	1	-4	-4	1	2	-4	-4	1	1	-4	-4	3	0	-4	-3
14752	1	0	-4	-4	3	3	-3	-3	2	0	-4	-4	3	0	-3	-4	3	3	-1	-4	3	3	-4	-1	3	1	-3	-4	3	1	-4	-3
14784	1	3	-3	-4	1	3	-4	-3	3	3	-2	-4	-1	3	-3	-4	-1	3	-4	-3	-3	3	-1	-4	-3	3	-4	-1	-3	1	-3	-4
14816	-3	1	-4	-3	-4	2	-2	-3	-2	3	-3	-4	-2	3	-4	-3	-4	3	-1	-3	-3	3	-4	-2	-3	2	-3	-4	-4	3	-3	-1
14848	-3	2	-4	-3	-4	1	-3	-3	-3	3	-2	-4	-4	2	-3	-2	-4	3	-2	-3	-4	3	-3	-2	-4	2	-3	-3	-3	2	-2	-4
14880	-4	1	-4	0	-1	2	-4	-4	-3	2	-4	-2	-2	1	-4	-4	-4	2	-1	-4	-4	2	-4	-1	-4	1	-2	-4	0	3	-3	-4
14912	-4	1	-4	-2	0	3	-4	-3	0	1	-4	-4	-1	1	-4	-4	0	2	-4	-4	-3	3	-4	0	-3	3	-3	-3	-4	2	-4	0
14944	-2	3	-2	-4	-4	3	-3	0	-4	1	-1	-4	-2	3	-4	-2	-4	1	-4	-1	-2	2	-3	-4	-4	3	-2	-2	-2	2	-4	-3
14976	1	-4	-3	-3	3	-2	-2	-4	3	-2	-4	-3	3	-3	-2	-4	3	-3	-4	-2	3	-2	-4	-2	3	-4	-2	-3	3	-4	-3	-2
15008	2	-3	-3	-4	2	-3	-4	-3	3	-4	-2	-2	2	-4	-3	-3	2	-2	-3	-4	2	-2	-4	-3	2	-3	-2	-4	2	-1	-4	-4
15040	2	-4	-1	-4	2	-3	-4	-2	2	-4	-4	-1	3	-3	-4	0	1	-2	-4	-4	1	-4	-2	-4	1	-4	-4	-2	3	-4	-3	0
15072	1	-1	-4	-4	1	-4	-1	-4	1	-4	-4	-1	2	-4	-2	-3	2	-4	-3	-2	0	-3	-3	-4	0	-3	-4	-3	1	-4	-4	0
15104	0	-4	-3	-3	0	-1	-4	-4	3	-3	-3	-3	2	-4	-4	0	0	-2	-4	-4	0	-4	-2	-4	0	-4	-1	-4	0	-4	-4	-2
15136	0	-4	-4	-1	3	-2	-3	-4	3	-1	-3	-4	3	-1	-4	-3	3	-3	-1	-4	3	-3	-4	-1	3	-4	-1	-3	3	-4	-3	-1
15168	1	-3	-3	-4	1	-3	-4	-3	-2	-2	-3	-4	-4	-3	-3	0	-1	-3	-3	-4	-1	-3	-4	-3	-2	-2	-4	-3	-4	-4	-1	-1
15200	-1	-4	-3	-3	-2	-3	-2	-4	-3	-1	-3	-4	-3	-1	-4	-3	-2	-3	-4	-2	-3	-3	-1	-4	-3	-3	-4	-1	-3	-4	-1	-3
15232	-3	-4	-3	-1	-2	-3	-3	-4	-2	-3	-4	-3	-2	-4	-3	-3	-3	-2	-3	-4	-3	-2	-4	-3	-3	-3	-2	-4	-3	-3	-4	-2
15264	-2	-4	-2	-3	-3	-4	-2	-3	-4	-1	-3	-3	-3	-4	-3	-2	-2	-4	-3	-2	-4	-3	-1	-3	-4	-3	-3	-1	-4	-2	-3	-3
15296	-4	-3	-2	-3	-4	-3	-3	-2	-1	-2	-4	-4	-1	-4	-2	-4	-1	-4	-4	-2	-2	-1	-4	-4	-2	-4	-1	-4	-2	-4	-4	-1
15328	-4	-2	-2	-3	-3	-2	-2	-4	-4	-1	-2	-4	-4	-1	-4	-2	-4	-2	-1	-4	-4	-2	-4	-1	-4	0	-1	-4	-4	-4	-1	-2
15360	-4	-4	-2	-1	-3	-2	-4	-2	-1	-1	-4	-4	-4	-3	-2	-2	-1	-4	-1	-4	-1	-4	-4	-1	-2	0	-4	-4	-2	-4	-4	0
15392	-3	0	-3	-4	-4	0	-4	-1	-3	0	-4	-3	-3	-3	-3	-3	-1	0	-4	-4	-3	-3	-4	0	-4	0	-2	-4	-3	-4	-2	-2
15424	-4	0	-4	-2	-4	-2	-4	0	-1	-4	-4	0	-3	-4	-3	0	-4	-4	-2	0	-4	-2	-3	-2	-4	-1	-4	0	-4	0	-3	-3
15456	-4	-1	-1	-4	-4	-1	-4	-1	-4	-4	-1	0	-4	3	3	3	3	-4	3	3	-4	-4	3	0	-4	-4	3	2	-4	-4	2	2
15488	-3	-4	3	3	-4	-4	1	3	-4	-4	3	3	-4	-4	3	1	-4	-4	2	3	-4	-3	3	3	3	3	-4	3	-4	3	-3	3
15520	-4	2	-4	3	-4	2	-4	2	-4	3	-4	1	-4	1	-4	3	-4	3	-4	2	-3	3	-4	3	-4	3	-4	3	3	-4	-4	2
15552	2	-4	-4	2	2	-4	-4	3	1	-4	-4	3	0	-4	-4	3	3	-4	-3	3	3	-3	-4	3	3	-4	-4	1	3	-4	-4	3
15584	-4	-4	-4	3	-4	-4	-1	3	-4	-3	-4	2	-4	-2	-4	3	-4	-3	-4	1	-3	-4	-4	1	-4	-3	-3	3	-4	-4	-4	1
15616	-4	-4	-3	3	-3	-4	-4	3	-3	-3	-4	3	-4	-4	-4	2	-4	-4	-3	1	-2	-4	-4	2	-1	-4	-4	3	-2	-4	-4	3
15648	-4	-4	-2	3	-4	-4	0	3	-3	-4	-4	2	-4	0	-4	3	-4	-1	-4	3	-4	-2	-4	2	-3	-4	-3	3	-4	-3	-4	3
15680	-4	-4	-3	2	-4	-4	-2	2	3	3	3	-4	-3	3	3	-4	-4	2	2	-4	-4	3	3	-4	-4	3	0	-4	-4	3	2	-4
15712	-4	3	1	-4	-4	2	3	-4	-4	1	3	-4	-4	3	3	-3	3	-4	2	-4	3	-3	3	-4	3	-4	3	-3	1	-4	3	-4
15744	2	-4	2	-4	3	-4	0	-4	3	-4	1	-4	3	-4	3	-4	0	-4	3	-4	2	-4	3	-4	-4	-3	0	-4	-4	-3	1	-4
15776	-4	-4	3	-4	-4	-4	1	-4	-4	-4	2	-4	-4	-4	3	-3	-4	-4	0	-3	-4	-3	3	-4	-4	-4	2	-2	-3	-4	3	-3
15808	-3	-4	3	-4	-4	-4	0	-4	-1	-4	3	-4	-2	-4	3	-4	-3	-4	2	-4	-4	-4	3	-1	-3	-4	1	-4	-3	-4	0	-4
15840	-4	-3	3	-3	-4	-2	3	-4	-4	-3	2	-4	-3	-3	3	-4	-4	0	3	-4	-4	-4	3	-2	-4	-4	2	-3	-4	-4	1	-3
15872	-2	-4	2	-4	-4	-2	2	-4	-4	-1	3	-4	3	3	-3	-4	3	3	-4	-3	1	3	-4	-4	3	3	-4	-4	2	3	-4	-4
15904	2	2	-4	-4	3	1	-4	-4	3	0	-4	-4	3	2	-4	-4	-3	3	-3	-4	0	3	-4	-4	-3	3	-4	-3	-4	3	-4	-4
15936	-4	1	-4	-4	-4	3	-3	-4	-4	3	-4	-3	-4	3	-3	-3	-3	3	-4	-4	-4	2	-2	-4	-2	3	-4	-4	-4	3	-2	-4
15968	-3	2	-4	-4	-3	1	-4	-4	-4	2	-4	-4	-4	2	-4	-2	-4	3	-4	-2	-4	2	-3	-4	-4	2	-4	-3	-2	2	-4	-4
16000	-4	3	-1	-4	-4	3	-4	0	-4	3	-4	-1	-4	1	-3	-4	-4	1	-4	-3	-1	3	-4	-4	3	-3	-3	-4	3	-3	-4	-3
16032	3	-2	-4	-4	3	-4	-3	-3	3	-4	-4	-2	2	-4	-4	-4	3	-3	-4	-4	3	-4	-3	-4	3	-4	-4	-3	3	-1	-4	-4
16064	1	-4	-4	-4	3	-4	-1	-4	2	-2	-4	-4	3	-4	-4	0	3	-4	-4	-1	0	-4	-4	-4	1	-3	-4	-4	0	-3	-4	-4
16096	1	-4	-3	-4	3	-4	-2	-4	1	-4	-4	-3	2	-3	-4	-4	0	-4	-3	-4	2	-4	-2	-4	2	-4	-3	-4	2	-4	-4	-3
16128	0	-4	-4	-3	2	-4	-4	-2	3	-4	-4	-4	-4	-1	-3	-4	-4	0	-4	-3	-4	-1	-4	-3	-2	-4	-2	-4	-4	-3	-1	-4
16160	-4	-3	-4	0	-4	-4	-4	-4	-4	-3	-4	-1	-3	-4	-4	-4	-4	-3	-4	-4	-4	-2	-2	-4	-3	-3	-3	-4	-4	-4	-4	-3

FIG 18(10)

16192	-2	-4	-4	-4	-3	-3	-4	-3	-4	-2	-4	-4	-4	-4	-2	-4	-4	-4	-4	-2	-3	-4	-3	-3	-2	-4	-4	-2	-3	-3	-4	-4
16224	-4	-4	-1	-3	-3	-4	-3	-4	-3	-4	-4	-3	-4	-4	-3	0	-4	-4	-3	-1	-4	-3	-3	-4	-4	-3	-3	-3	-4	-3	-4	-3
16256	-4	-4	-3	-3	-1	-4	-4	-4	-4	-1	-4	-4	-4	-4	-1	-4	-4	-4	-4	-1	-4	0	-4	-4	-4	-4	-4	0	-1	-3	-4	-4
16288	-2	-3	-4	-4	-1	-4	-3	-4	-2	-4	-3	-4	-4	-2	-4	-2	-2	-4	-4	-3	-1	-4	-4	-3	-3	-2	-4	-4	-3	0	-4	-4
16320	-3	-4	-2	-4	-3	-1	-4	-4	-3	-4	-4	-2	-3	-4	-1	-4	-3	-4	-4	0	-3	-4	-4	-1	-4	-2	-3	-4	-4	-2	-4	-3
16352	-4	-4	-3	-4	-4	-3	-2	-4	-4	-3	-4	-2	-2	-2	-4	-4	-4	0	-3	-4	-4	-4	-2	-3	-4	-4	-3	-2	-4	-4	-2	-2

FIG 18(11)

1970 egroups[a][b][c][d] 1982a 1982b 1982c 1982d 1901 1982e 1982f 1982g 1982h

a=0, b=0

c/d (1980)	0	1	2	3	4	5	6	7
0 (1972a)	0x021F063D	0x021F0C3D	0x021F113D	0x021F213D	0x021F233D	0x0217071A	0x02170B1A	0x0217001A
1 (1972b)	0x021F353D	0x021F1D3D	0x021F3B3D	0x021F193D	0x021F183D	0x02170C1A	0x0217181A	0x0217081A
2 (1972c)	0x021F103D	0x021F3A3D	0x021F3C3D	0x02102E49	0x02104249	0x02081C2D	0x0217191A	0x0217101A
3 (1972d)	0x021F203D	0x021F153D	0x02102D49	0x02100549	0x02104849	0x0208012D	0x0208122D	0x0217011A
4 (1972e)	0x021B0B1D	0x021B061D	0x020C2C32	0x020C3132	0x01F41925	0x0208262D	0x0208252D	0x0217111A
5 (1972f)	0x021B031D	0x021B191D	0x020C1F32	0x020C0232	0x02040E1B	0x0204011B	0x0204171B	0x02130409
6 (1972g)	0x021B041D	0x021B181D	0x021B081D	0x020C1E32	0x0204081B	0x0204111B	0x02130209	0x02130709
7 (1972h)	0x021B021D	0x021B051D	0x021B171D	0x021B0F1D	0x02130009	0x02130609	0x02130109	0x02130509

a=0, b=1 1902

c/d	0	1	2	3	4	5	6	7
0	0x021F093D	0x021F1C3D	0x021F373D	0x021F073D	0x021F053D	0x0217041A	0x0217021A	0x0217171A
1	0x021F1A3D	0x021F1B3D	0x02101F49	0x02101C49	0x02100149	0x0208202D	0x0208112D	0x0217061A
2	0x021F363D	0x02101E49	0x02103149	0x02002B3F	0x02000B3F	0x01F81E30	0x02080D2D	0x02081A2D
3	0x021F043D	0x02101B49	0x0200293F	0x02003E3F	0x01F01F32	0x01F82E30	0x01F81D30	0x02080C2D
4	0x021B001D	0x020C0032	0x01FC0633	0x01EC1E2C	0x01E41E23	0x01E81229	0x01F80530	0x0208162D
5	0x021B011D	0x020C1332	0x01FC2133	0x01FC1233	0x01E40E23	0x01F41725	0x01F40925	0x0204131B
6	0x021B141D	0x020C1632	0x020C0832	0x01FC1F33	0x01F41C25	0x01F41A25	0x0204191B	0x0204151B
7	0x021B071D	0x021B121D	0x020C1532	0x020C1232	0x0204101B	0x0204161B	0x02040D1B	0x02130809

FIG 19(1)

1903

a=0, b=2

c/d	0	1	2	3	4	5	6	7
0	0x021F0F3D	0x021F343D	0x021F283D	0x02102B49	0x02103F49	0x0208192D	0x0217151A	0x0217031A
1	0x021F333D	0x02101D49	0x02104349	0x0200283F	0x0200093F	0x01F81B30	0x0208142D	0x02081B2D
2	0x021F0A3D	0x02102649	0x02003A3F	0x01F00332	0x01F01932	0x01E81429	0x01F80D30	0x02081F2D
3	0x02102A49	0x0200263F	0x01F00232	0x01E0242B	0x01E0012B	0x01D81B22	0x01E81A29	0x01F81A30
4	0x020C2A32	0x01FC0533	0x01EC182C	0x01DC0226	0x01E02A2B	0x01D80122	0x01E80029	0x01F80430
5	0x020C1D32	0x01FC1E33	0x01EC2B2C	0x01DC2126	0x01D4101D	0x01D4141D	0x01E40623	0x01F40D25
6	0x021B1B1D	0x020C0B32	0x01FC2D33	0x01EC012C	0x01E40B23	0x01E41A23	0x01F41F25	0x0204091B
7	0x021B131D	0x020C1432	0x020C0632	0x01FC1D33	0x01F40625	0x01F40525	0x02040A1B	0x02040F1B

1904

a=0, b=3

c/d	0	1	2	3	4	5	6	7
0	0x021F1F3D	0x021F023D	0x02102949	0x02104749	0x02104449	0x02082B2D	0x0208282D	0x0217141A
1	0x021F003D	0x02101849	0x0200243F	0x02002C3F	0x01F01E32	0x01F80030	0x01F81830	0x0208102D
2	0x02102849	0x0200233F	0x01F00132	0x01E0232B	0x01E0002B	0x01D81A22	0x01E82729	0x01F81730
3	0x02104649	0x02000C3F	0x01E0222B	0x01D01B1E	0x01D01D1E	0x01C8191A	0x01D81922	0x01F82430
4	0x020C2D32	0x01EC1D2C	0x01DC0026	0x01CC181B	0x01BC050A	0x01C80A1A	0x01D80022	0x01E81529
5	0x020C2F32	0x01FC2633	0x01DC1F26	0x01CC191B	0x01C41718	0x01C40718	0x01D40E1D	0x01F41525
6	0x020C1B32	0x01FC1B33	0x01EC002C	0x01DC1E26	0x01D41B1D	0x01D40B1D	0x01E40923	0x01F40325
7	0x021B1C1D	0x020C1032	0x01FC1A33	0x01FC1833	0x01E41723	0x01F40825	0x01F41625	0x0204051B

FIG 19(2)

1905

a=0, b=4

c/d	0	1	2	3	4	5	6	7
0	0x021F223D	0x021F013D	0x02103E49	0x02103749	0x02002F3F	0x02081E2D	0x0208152D	0x0217131A
1	0x021F393D	0x02104549	0x0200073F	0x01F01D32	0x01F00032	0x01E81129	0x01F82A30	0x02082A2D
2	0x02103C49	0x0200063F	0x01F01532	0x01E0272B	0x01E0292B	0x01D81F22	0x01E81929	0x01F80230
3	0x02102C49	0x01F01C32	0x01E0262B	0x01D0111E	0x01C0040A	0x01C8081A	0x01D81D22	0x01E81029
4	0x01FC2533	0x01EC032C	0x01DC0626	0x01BC030A	0x01C0090A	0x01B80209	0x01D82122	0x01E82829
5	0x020C1A32	0x01EC1B2C	0x01DC2426	0x01CC011B	0x01B40108	0x01C41518	0x01D4001D	0x01E40123
6	0x020C2832	0x01FC0433	0x01EC152C	0x01DC2326	0x01D4021D	0x01D41C1D	0x01E42123	0x01F40125
7	0x021B161D	0x020C2E32	0x01FC0333	0x01EC1A2C	0x01E41F23	0x01E41923	0x01F41D25	0x0204121B

1906

a=0, b=5

c/d	0	1	2	3	4	5	6	7
0	0x021D041A	0x021D181A	0x020E1D2D	0x020E292D	0x020E152D	0x02060F18	0x02060018	0x02150408
1	0x021D171A	0x020E0E2D	0x01FE1730	0x01FE0730	0x01EE1428	0x01F61B22	0x01F61622	0x02060C18
2	0x020E1B2D	0x01FE1630	0x01EE2728	0x01DE1F23	0x01DE2123	0x01D6141A	0x01E6021E	0x01F60822
3	0x020E272D	0x01FE0130	0x01DE1E23	0x01CE1A1B	0x01CE101B	0x01C61617	0x01D6131A	0x01F61C22
4	0x020A0D1B	0x01EA1122	0x01DA1B1D	0x01CA0518	0x01BE0209	0x01C61417	0x01D6171A	0x01E6141E
5	0x020A151B	0x01FA1D24	0x01DA191D	0x01CA1618	0x01C20F15	0x01C20715	0x01D21015	0x01F21416
6	0x020A101B	0x01FA1324	0x01EA2122	0x01DA181D	0x01D20815	0x01D21415	0x01E21119	0x01F21016
7	0x02190709	0x020A081B	0x01FA1224	0x01FA1824	0x01E21519	0x01F20016	0x01F20816	0x02020A0B

FIG 19(3)

1907

a=0, b=6

c/d	0	1	2	3	4	5	6	7
0	0x021D0E1A	0x021D121A	0x021D0F1A	0x020E1A2D	0x020E242D	0x02061418	0x02150208	0x02150108
1	0x021D111A	0x020E132D	0x020E102D	0x01FE1530	0x01FE0230	0x01F60F22	0x02060D18	0x02060318
2	0x021D091A	0x020E062D	0x01FE2A30	0x01EE2528	0x01EE1128	0x01E6081E	0x01F62122	0x02060218
3	0x020E192D	0x01FE1330	0x01EE2328	0x01DE1C23	0x01DE2023	0x01D60C1A	0x01E6071E	0x01F60E22
4	0x020A191B	0x01FA0024	0x01EA0E22	0x01DA1A1D	0x01DE0A23	0x01D6161A	0x01E6061E	0x01F60422
5	0x020A0F1B	0x01FA1024	0x01EA1F22	0x01DA161D	0x01D21115	0x01D20715	0x01E20519	0x01F20616
6	0x02190109	0x020A041B	0x01FA0524	0x01EA1E22	0x01E21819	0x01E21719	0x01F21216	0x0202070B
7	0x02190609	0x020A0C1B	0x020A011B	0x01FA0B24	0x01F20C16	0x01F21316	0x0202090B	0x0202020B

1908

a=0, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021D031A	0x021D061A	0x021D101A	0x021D161A	0x021D151A	0x02150308	0x02150508	0x02150708
1	0x021D051A	0x021D071A	0x020E122D	0x020E0C2D	0x020E262D	0x02060918	0x02061318	0x02150008
2	0x021D0B1A	0x020E112D	0x020E2B2D	0x01FE1130	0x01FE0030	0x01F60322	0x02061018	0x02061518
3	0x021D141A	0x020E092D	0x01FE1030	0x01FE1B30	0x01EE1328	0x01F61022	0x01F61122	0x02060A18
4	0x02190309	0x020A141B	0x01FA0224	0x01EA0F22	0x01E20619	0x01E6091E	0x01F60022	0x02061218
5	0x02190509	0x020A091B	0x01FA0E24	0x01FA0C24	0x01E20819	0x01F20E16	0x01F20516	0x0202010B
6	0x02190409	0x020A0B1B	0x020A161B	0x01FA0D24	0x01F20A16	0x01F20416	0x0202050B	0x0202030B
7	0x02190209	0x02190809	0x020A0A1B	0x020A051B	0x0202000B	0x0202060B	0x0202040B	0x02110001

FIG 19(4)

1909

a=1, b=0

c/d	0	1	2	3	4	5	6	7
0	0x021F083D	0x021F173D	0x021F2F3D	0x021F323D	0x021F313D	0x0217051A	0x0217121A	0x0217091A
1	0x021F163D	0x021F123D	0x02101949	0x02100E49	0x02104149	0x0208082D	0x02080B2D	0x0217161A
2	0x021F2D3D	0x02101749	0x02103D49	0x0200213F	0x0200033F	0x01F82030	0x0208212D	0x02080A2D
3	0x021F303D	0x02100D49	0x0200203F	0x0200183F	0x01F01A32	0x01F80B30	0x01F81330	0x0208242D
4	0x021B111D	0x020C2B32	0x01FC1533	0x01EC192C	0x01E82029	0x01E80E29	0x01F81530	0x02080F2D
5	0x021B101D	0x020C1C32	0x01FC1733	0x01FC0833	0x01E40F23	0x01F42125	0x01F40F25	0x0204041B
6	0x021B0E1D	0x020C1132	0x020C2132	0x01FC1633	0x01F40A25	0x01F41325	0x02040B1B	0x0204181B
7	0x021B0A1D	0x021B091D	0x020C0332	0x020C2232	0x0204001B	0x02041A1B	0x0204061B	0x02130309

1910

a=1, b=1

c/d	0	1	2	3	4	5	6	7
0	0x021F143D	0x021F0E3D	0x02101549	0x02100C49	0x02103B49	0x0208232D	0x0208062D	0x02170A1A
1	0x021F0B3D	0x02103949	0x02003D3F	0x0200323F	0x0200083F	0x01F82630	0x01F82D30	0x0208272D
2	0x02101449	0x02003B3F	0x01F01832	0x01F03132	0x01E01F2B	0x01E80F29	0x01E80C29	0x01F82330
3	0x02100B49	0x0200313F	0x01F01B32	0x01E0022B	0x01D01A1E	0x01D82022	0x01E82429	0x01F82530
4	0x020C2732	0x01FC0133	0x01DC1B26	0x01CC171B	0x01D0081E	0x01C8181A	0x01D81622	0x01F81230
5	0x020C2532	0x01FC2833	0x01EC212C	0x01DC2026	0x01C41118	0x01D41A1D	0x01E40323	0x01F41825
6	0x020C0F32	0x01FC3033	0x01EC172C	0x01EC072C	0x01D4181D	0x01E40A23	0x01E40423	0x01F40225
7	0x021B151D	0x020C2632	0x01FC2F33	0x01FC2733	0x01F40C25	0x01F41225	0x01F40E25	0x0204071B

FIG 19(5)

1911

a=1, b=2

c/d	0	1	2	3	4	5	6	7
0	0x021F2B3D	0x02101349	0x02102F49	0x02001F3F	0x0200013F	0x01F81130	0x0208172D	0x0208052D
1	0x02101249	0x0200393F	0x01F01632	0x01F02932	0x01E01D2B	0x01E81829	0x01E80B29	0x01F82B30
2	0x02102749	0x01F01432	0x01E0202B	0x01D0021E	0x01C0010A	0x01C8011A	0x01D81822	0x01E80A29
3	0x02001E3F	0x01F02032	0x01D01C1E	0x01B00B0C	0x01A00B0C	0x01A80B0C	0x01C8171A	0x01E80D29
4	0x01FC0033	0x01DC1926	0x01BC000A	0x019C0B0C	0x01900607	0x0198070C	0x01B80609	0x01D81422
5	0x01FC1433	0x01EC0F2C	0x01CC121B	0x01AC060C	0x01940A0C	0x01A4040C	0x01C40F18	0x01E40823
6	0x020C1932	0x01EC162C	0x01DC1D26	0x01CC0E1B	0x01B40408	0x01C40D18	0x01D4191D	0x01E40C23
7	0x020C0E32	0x01FC2E33	0x01EC142C	0x01EC022C	0x01D4161D	0x01E40D23	0x01E41223	0x01F42025

1912

a=1, b=3

c/d	0	1	2	3	4	5	6	7
0	0x021F2E3D	0x02100949	0x02001D3F	0x0200373F	0x01F01332	0x01F82730	0x01F82F30	0x02082C2D
1	0x02100849	0x02002E3F	0x01F03032	0x01E00E2B	0x01D0191E	0x01D81E22	0x01E81F29	0x01F82130
2	0x02001C3F	0x01F02332	0x01D00F1E	0x01B0080C	0x01A0070C	0x01A8050C	0x01C80C1A	0x01E81729
3	0x0200303F	0x01E0032B	0x01B0040C	0x01800506	0x01700506	0x01780506	0x01A8080C	0x01D80222
4	0x01EC132C	0x01CC161B	0x019C080C	0x016C0506	0x01600809	0x01680206	0x0198020C	0x01C8161A
5	0x01FC2433	0x01DC0326	0x01AC070C	0x017C0306	0x01640406	0x01740306	0x01A40A0C	0x01D4031D
6	0x01FC1333	0x01EC1F2C	0x01CC0A1B	0x01AC020C	0x01940B0C	0x01A4020C	0x01C40618	0x01E41323
7	0x020C0732	0x01FC2333	0x01EC1C2C	0x01DC2226	0x01C40518	0x01D4061D	0x01E41023	0x01F41B25

FIG 19(6)

1913

a=1, b=4

c/d	0	1	2	3	4	5	6	7
0	0x021F2C3D	0x02103849	0x0200003F	0x01F01232	0x01F02F32	0x01E81329	0x01F82C30	0x0208222D
1	0x02103649	0x02002D3F	0x01E01C2B	0x01D0181E	0x01D0071E	0x01C8151A	0x01D81322	0x01F80730
2	0x02003C3F	0x01E01A2B	0x01C0000A	0x01A0080C	0x01900507	0x0198090C	0x01B80809	0x01D81122
3	0x01F01132	0x01D0171E	0x01A0060C	0x01700306	0x01600709	0x01680106	0x0198080C	0x01C8141A
4	0x01EC2A2C	0x01CC051B	0x018C0307	0x015C0209	0x01500607	0x01580508	0x01880506	0x01C8031A
5	0x01EC122C	0x01CC151B	0x019C060C	0x016C0006	0x01540508	0x01640306	0x0194010C	0x01C40A18
6	0x01FC3133	0x01DC1826	0x01BC070A	0x019C030C	0x01840406	0x0194080C	0x01B40208	0x01D4151D
7	0x020C2432	0x01FC2233	0x01DC1626	0x01CC141B	0x01C40918	0x01C41318	0x01D4131D	0x01F41425

1914

a=1, b=5

c/d	0	1	2	3	4	5	6	7
0	0x021D0D1A	0x020E052D	0x01FE0F30	0x01FE1C30	0x01EE1028	0x01F61222	0x01F60B22	0x02061118
1	0x020E042D	0x01FE2030	0x01EE1228	0x01DE1B23	0x01CE141B	0x01D6181A	0x01E6031E	0x01F61A22
2	0x01FE0E30	0x01EE0828	0x01CE011B	0x01AE010C	0x019E090C	0x01A6020C	0x01C60017	0x01E6121E
3	0x01FE1830	0x01DE1823	0x01AE000C	0x017E0206	0x016E0406	0x01760106	0x01A60B0C	0x01D6021A
4	0x01EA0D22	0x01CA1418	0x019A090C	0x016A0406	0x015E0309	0x01660406	0x0196080C	0x01C61017
5	0x01FA1124	0x01DA0B1D	0x01AA0B0C	0x017A0006	0x01620006	0x01720306	0x01A20A0C	0x01D20215
6	0x01FA0A24	0x01EA0422	0x01CA1718	0x01AA090C	0x0192020C	0x01A2090C	0x01C20315	0x01E21419
7	0x020A031B	0x01FA1A24	0x01EA0022	0x01DA061D	0x01C21215	0x01D20E15	0x01E21019	0x01F21516

FIG 19(7)

1915

a=1, b=6

c/d	0	1	2	3	4	5	6	7
0	0x021D0C1A	0x020E0D2D	0x020E172D	0x01FE0D30	0x01FE2F30	0x01F60A22	0x02060718	0x02060118
1	0x020E0B2D	0x01FE2D30	0x01EE0F28	0x01EE2428	0x01DE1A23	0x01E61C1E	0x01E6051E	0x01F62022
2	0x020E142D	0x01EE0E28	0x01DE1D23	0x01CE181B	0x01BE0809	0x01C61217	0x01D6121A	0x01E6131E
3	0x01FE0C30	0x01EE2128	0x01CE151B	0x01AE050C	0x019E070C	0x01A6080C	0x01C60A17	0x01E6181E
4	0x01FA2324	0x01DA151D	0x01BA0708	0x019A070C	0x018E0407	0x0196050C	0x01B60607	0x01D6101A
5	0x01FA0924	0x01EA1A22	0x01CA0F18	0x01AA020C	0x0192090C	0x01A2060C	0x01C20015	0x01E20419
6	0x020A1A1B	0x01EA0C22	0x01DA171D	0x01CA0C18	0x01B20007	0x01C20B15	0x01D20915	0x01E20C19
7	0x020A061B	0x01FA2224	0x01EA0B22	0x01EA1722	0x01D20315	0x01E20319	0x01E20D19	0x01F21116

1916

a=1, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021D081A	0x021D021A	0x020E0A2D	0x020E032D	0x020E222D	0x02060E18	0x02060518	0x02150608
1	0x021D001A	0x020E232D	0x01FE2C30	0x01FE1F30	0x01FE1230	0x01F60722	0x01F60622	0x02060618
2	0x020E0F2D	0x01FE2B30	0x01EE0D28	0x01EE1928	0x01DE1923	0x01E60F1E	0x01E6041E	0x01F60222
3	0x020E022D	0x01FE1D30	0x01EE1728	0x01DE0423	0x01CE121B	0x01D6001A	0x01E60C1E	0x01F61822
4	0x020A111B	0x01FA1924	0x01DA141D	0x01CA1118	0x01CE081B	0x01C61517	0x01D60F1A	0x01F61E22
5	0x020A021B	0x01FA1724	0x01EA1422	0x01DA021D	0x01C20A15	0x01D21315	0x01E21319	0x01F20916
6	0x020A071B	0x01FA0124	0x01EA0A22	0x01EA1222	0x01D20B15	0x01E20119	0x01E21619	0x01F20316
7	0x02190009	0x020A121B	0x01FA2024	0x01FA1524	0x01F20F16	0x01F20B16	0x01F20D16	0x0202080B

FIG 19(8)

1917

a=2, b=0

c/d	0	1	2	3	4	5	6	7
0	0x021F0D3D	0x021F293D	0x021F133D	0x02102549	0x02103549	0x0208042D	0x02170D1A	0x02170F1A
1	0x021F273D	0x02101149	0x02101A49	0x02001A3F	0x0200383F	0x01F81030	0x0208002D	0x0208032D
2	0x021F033D	0x02101649	0x02002A3F	0x01F02E32	0x01F00F32	0x01E82229	0x01F81C30	0x0208092D
3	0x02102449	0x0200193F	0x01F02D32	0x01E0172B	0x01E01E2B	0x01D80E22	0x01E82529	0x01F80F30
4	0x020C2332	0x01FC2C33	0x01EC112C	0x01DC1A26	0x01DC0126	0x01D81522	0x01E82629	0x01F81430
5	0x020C1832	0x01FC1133	0x01EC292C	0x01DC1326	0x01D4071D	0x01D40A1D	0x01E40723	0x01F40425
6	0x021B1A1D	0x020C0D32	0x01FC2033	0x01EC282C	0x01E41123	0x01E40023	0x01F40025	0x0204021B
7	0x021B0D1D	0x020C0C32	0x020C0932	0x01FC1033	0x01F40B25	0x01F41E25	0x0204031B	0x02040C1B

1918

a=2, b=1

c/d	0	1	2	3	4	5	6	7
0	0x021F253D	0x02101049	0x02100A49	0x0200173F	0x0200363F	0x01F80E30	0x0208182D	0x0208292D
1	0x02100F49	0x0200343F	0x01F00D32	0x01F01732	0x01E0162B	0x01E80829	0x01E80629	0x01F82830
2	0x02100749	0x01F00B32	0x01E01B2B	0x01D0101E	0x01C0060A	0x01C80F1A	0x01D81222	0x01E80229
3	0x0200163F	0x01F01032	0x01D00E1E	0x01B0060C	0x01A0040C	0x01A8040C	0x01C80E1A	0x01E80729
4	0x01FC2B33	0x01DC1126	0x01BC060A	0x019C050C	0x018C0107	0x0198050C	0x01B80709	0x01D80D22
5	0x01FC0F33	0x01EC0E2C	0x01CC0C1B	0x01AC040C	0x0194000C	0x01A40B0C	0x01C40818	0x01E41523
6	0x020C0532	0x01EC0D2C	0x01DC1726	0x01CC091B	0x01B40308	0x01C40C18	0x01D4111D	0x01E42023
7	0x020C0A32	0x01FC2A33	0x01EC0C2C	0x01EC092C	0x01D4011D	0x01E41B23	0x01E40523	0x01F42325

FIG 19(9)

1919

a=2, b=2

c/d	0	1	2	3	4	5	6	7
0	0x021F383D	0x02100449	0x0200273F	0x01F02C32	0x01F00E32	0x01E82329	0x01F81930	0x02081D2D
1	0x02100049	0x01F00932	0x01E0192B	0x01D0091E	0x01C0050A	0x01C8061A	0x01D81022	0x01E80529
2	0x0200253F	0x01E0182B	0x01C0080A	0x01900407	0x01800406	0x01880306	0x01B80009	0x01D80F22
3	0x01F02B32	0x01D0061E	0x01900107	0x01600009	0x01500507	0x01580708	0x01880106	0x01C8051A
4	0x01EC102C	0x01BC040A	0x017C0506	0x014C0306	0x01290001	0x01480506	0x01780306	0x01B80409
5	0x01EC272C	0x01CC041B	0x018C0207	0x015C0609	0x01440306	0x01540308	0x01840206	0x01C40118
6	0x01FC1C33	0x01DC1426	0x01BC080A	0x018C0007	0x01740406	0x01840006	0x01B40608	0x01D4041D
7	0x020C3032	0x01EC0A2C	0x01DC1226	0x01CC021B	0x01B40708	0x01C40018	0x01D4051D	0x01E41823

1920

a=2, b=3

c/d	0	1	2	3	4	5	6	7
0	0x02102349	0x0200153F	0x01F02A32	0x01E0142B	0x01E0132B	0x01D80C22	0x01E82129	0x01F80C30
1	0x0200143F	0x01F00432	0x01D0011E	0x01B0020C	0x01A0010C	0x01A8010C	0x01C8021A	0x01E80329
2	0x01F02832	0x01D0001E	0x01900307	0x01600309	0x01500307	0x01580108	0x01880406	0x01C8001A
3	0x01E0122B	0x01B0000C	0x01600109	0x00E90001	0x00890001	0x00EA0001	0x01580008	0x01A80A0C
4	0x01DC0F26	0x019C020C	0x014C0006	0x008A0001	0x00590001	0x008B0001	0x01480406	0x0198000C
5	0x01DC1026	0x01AC0B0C	0x015C0109	0x00EB0001	0x008C0001	0x00EC0001	0x01540008	0x01A4000C
6	0x01EC262C	0x01CC001B	0x018C0607	0x015C0809	0x01440106	0x01540608	0x01840506	0x01C41018
7	0x01FC0E33	0x01EC052C	0x01CC1A1B	0x01AC090C	0x0194090C	0x01A4080C	0x01C40218	0x01E41C23

FIG 19(10)

1921

a=2, b=4

c/d	0	1	2	3	4	5	6	7
0	0x02103449	0x0200353F	0x01F00C32	0x01E0102B	0x01E0282B	0x01D80422	0x01E80129	0x01F82930
1	0x0200333F	0x01E0112B	0x01C0030A	0x01A00A0C	0x01900007	0x01980B0C	0x01B80309	0x01D80B22
2	0x01F00A32	0x01C0020A	0x01800106	0x01500407	0x012A0001	0x01480206	0x01780106	0x01B80109
3	0x01E00F2B	0x01A0090C	0x01500207	0x008D0001	0x005A0001	0x008E0001	0x01480306	0x01980A0C
4	0x01DC2526	0x018C0407	0x012B0001	0x005B0001	0x00410001	0x005C0001	0x012C0001	0x01880006
5	0x01DC0D26	0x019C0A0C	0x014C0106	0x008F0001	0x005D0001	0x00900001	0x01440406	0x0194040C
6	0x01EC0B2C	0x01BC020A	0x017C0106	0x014C0406	0x012D0001	0x01440206	0x01740106	0x01B40008
7	0x01FC2933	0x01DC0E26	0x01BC010A	0x019C090C	0x01840106	0x0194070C	0x01B40508	0x01D4081D

1922

a=2, b=5

c/d	0	1	2	3	4	5	6	7
0	0x020E182D	0x01FE0B30	0x01EE2028	0x01DE1723	0x01DE1423	0x01D6051A	0x01E6111E	0x01F61522
1	0x01FE0A30	0x01EE0028	0x01CE191B	0x01AE0A0C	0x019E080C	0x01A6050C	0x01C60717	0x01E61B1E
2	0x01EE1F28	0x01CE171B	0x018E0307	0x015E0809	0x014E0407	0x01560408	0x01860406	0x01C61317
3	0x01DE1623	0x01AE080C	0x015E0509	0x00ED0001	0x00910001	0x00EE0001	0x01560208	0x01A6040C
4	0x01DA111D	0x019A080C	0x014A0306	0x00920001	0x005E0001	0x00930001	0x01460306	0x0196070C
5	0x01DA131D	0x01AA070C	0x015A0609	0x00EF0001	0x00940001	0x00F00001	0x01520308	0x01A2010C
6	0x01EA1B22	0x01CA1518	0x018A0207	0x015A0409	0x01420406	0x01520008	0x01820506	0x01C20E15
7	0x01FA0724	0x01EA0122	0x01CA1318	0x01AA050C	0x0192050C	0x01A2030C	0x01C20D15	0x01E20019

FIG 19(11)

1923

a=2, b=6

c/d	0	1	2	3	4	5	6	7
0	0x021D131A	0x020E2C2D	0x01FE1A30	0x01EE1E28	0x01EE0C28	0x01E6171E	0x01F60522	0x02060B18
1	0x020E2A2D	0x01EE0B28	0x01DE1323	0x01CE0F1B	0x01BE0109	0x01C60E17	0x01D6031A	0x01E6011E
2	0x01FE1930	0x01DE1223	0x01BE0409	0x018E0207	0x017E0406	0x01860206	0x01B60007	0x01D6041A
3	0x01EE1D28	0x01CE0E1B	0x018E0107	0x015E0109	0x014E0307	0x01560608	0x01860106	0x01C60217
4	0x01EA0922	0x01BA0108	0x017A0506	0x014A0206	0x012E0001	0x01460506	0x01760306	0x01B60107
5	0x01EA1922	0x01CA0D18	0x018A0607	0x015A0109	0x01420106	0x01520608	0x01820306	0x01C20115
6	0x01FA1424	0x01DA101D	0x01BA0608	0x018A0407	0x01720406	0x01820006	0x01B20307	0x01D21215
7	0x020A181B	0x01EA0822	0x01DA0F1D	0x01CA0B18	0x01B20107	0x01C20515	0x01D20D15	0x01E20E19

1924

a=2, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021D0A1A	0x020E082D	0x020E282D	0x01FE0930	0x01FE2730	0x01F61722	0x02060418	0x02060818
1	0x020E072D	0x01FE2830	0x01EE0A28	0x01EE2628	0x01DE1523	0x01E6101E	0x01E61A1E	0x01F60122
2	0x020E252D	0x01EE0928	0x01DE0F23	0x01CE091B	0x01BE0009	0x01C60817	0x01D60B1A	0x01E60D1E
3	0x01FE0830	0x01EE2228	0x01CE051B	0x01AE030C	0x019E060C	0x01A6000C	0x01C60F17	0x01E6191E
4	0x01FA1F24	0x01DA121D	0x01BA0008	0x019A060C	0x018A0307	0x0196060C	0x01B60207	0x01D60E1A
5	0x01FA0624	0x01EA1D22	0x01CA0618	0x01AA010C	0x01920A0C	0x01A2000C	0x01C20215	0x01E20A19
6	0x020A131B	0x01EA0722	0x01DA0D1D	0x01CA0A18	0x01B20607	0x01C20415	0x01D20015	0x01E20719
7	0x020A0E1B	0x01FA0824	0x01EA0622	0x01EA1C22	0x01D20115	0x01E20F19	0x01E21219	0x01F20216

FIG 19(12)

1925

a=3, b=0

c/d	0	1	2	3	4	5	6	7
0	0x021F1E3D	0x021F2A3D	0x02102249	0x02103349	0x02104049	0x0208072D	0x0208132D	0x02170E1A
1	0x021F263D	0x02100649	0x0200123F	0x02001B3F	0x01F00832	0x01F80A30	0x01F81F30	0x0208022D
2	0x02102149	0x0200113F	0x01F02632	0x01E00D2B	0x01E0072B	0x01D80822	0x01E81D29	0x01F80930
3	0x02103249	0x0200133F	0x01E00C2B	0x01D0161E	0x01D00D1E	0x01C8131A	0x01D80722	0x01F82230
4	0x020C2932	0x01EC082C	0x01DC0726	0x01CC0D1B	0x01B80509	0x01C80D1A	0x01D80A22	0x01E80929
5	0x020C2032	0x01FC0B33	0x01DC0B26	0x01CC131B	0x01C40E18	0x01C41218	0x01D40D1D	0x01F41025
6	0x020C1732	0x01FC0D33	0x01EC252C	0x01DC0A26	0x01D4121D	0x01D4091D	0x01E41D23	0x01F40725
7	0x021B0C1D	0x020C0432	0x01FC0C33	0x01FC0733	0x01E40223	0x01F42225	0x01F42425	0x0204141B

1926

a=3, b=1

c/d	0	1	2	3	4	5	6	7
0	0x021F243D	0x02100349	0x0200103F	0x02000A3F	0x01F00732	0x01F80330	0x01F80830	0x02080E2D
1	0x02100249	0x0200223F	0x01F02732	0x01E0252B	0x01D0151E	0x01D81C22	0x01E81E29	0x01F81630
2	0x02000F3F	0x01F02532	0x01D00C1E	0x01B0070C	0x01A0050C	0x01A8060C	0x01C80B1A	0x01E81B29
3	0x0200053F	0x01E0212B	0x01B0050C	0x01800306	0x01700006	0x01780406	0x01A8030C	0x01D81722
4	0x01EC062C	0x01CC111B	0x019C070C	0x016C0106	0x015C0009	0x01680006	0x0198060C	0x01C8121A
5	0x01FC0233	0x01DC1C26	0x01AC050C	0x017C0406	0x01640506	0x01740206	0x01A4010C	0x01D40F1D
6	0x01FC0A33	0x01EC232C	0x01CC0B1B	0x01AC030C	0x0194020C	0x01A4030C	0x01C41418	0x01E42223
7	0x020C0132	0x01FC1933	0x01EC222C	0x01DC1526	0x01C40B18	0x01D40C1D	0x01E41423	0x01F41125

FIG 19(13)

1927

a=3, b=2

c/d	0	1	2	3	4	5	6	7
0	0x02102049	0x02000E3F	0x01F02432	0x01E00B2B	0x01E0062B	0x01D80322	0x01E81C29	0x01F80630
1	0x02000D3F	0x01F02132	0x01D00B1E	0x01B0030C	0x01A0030C	0x01A8020C	0x01C8091A	0x01E81629
2	0x01F02232	0x01D00A1E	0x01900207	0x01600609	0x01500107	0x01580608	0x01880206	0x01C8071A
3	0x01E00A2B	0x01B0010C	0x01600509	0x00F10001	0x00950001	0x00F20001	0x01580308	0x01A8000C
4	0x01DC0526	0x019C040C	0x014C0206	0x00960001	0x005F0001	0x00970001	0x01480106	0x0198040C
5	0x01DC0926	0x01AC010C	0x015C0709	0x00F30001	0x00980001	0x00F40001	0x01540708	0x01A4060C
6	0x01EC242C	0x01CC081B	0x018C0507	0x015C0509	0x01440006	0x01540408	0x01840306	0x01C40418
7	0x01FC0933	0x01EC202C	0x01CC071B	0x01AC000C	0x0194030C	0x01A4050C	0x01C40318	0x01E41623

1928

a=3, b=3

c/d	0	1	2	3	4	5	6	7
0	0x02103049	0x0200043F	0x01E0092B	0x01D0141E	0x01D0051E	0x01C8101A	0x01D80622	0x01F80130
1	0x0200023F	0x01E0152B	0x01B00A0C	0x01800206	0x01700406	0x01780206	0x01A8090C	0x01D80922
2	0x01E0082B	0x01B0090C	0x01600409	0x00F50001	0x00990001	0x00F60001	0x01580408	0x01A8070C
3	0x01D0131E	0x01800006	0x00F70001	0x00420001	0x00210001	0x00430001	0x00F80001	0x01780006
4	0x01CC061B	0x016C0406	0x009A0001	0x00220001	0x00090001	0x00230001	0x009B0001	0x01680506
5	0x01CC101B	0x017C0206	0x00F90001	0x00440001	0x00240001	0x00450001	0x00FA0001	0x01740006
6	0x01DC0826	0x01AC0A0C	0x015C0409	0x00FB0001	0x009C0001	0x00FC0001	0x01540108	0x01A4070C
7	0x01FC3233	0x01DC0C26	0x01AC080C	0x017C0006	0x01640106	0x01740506	0x01A4090C	0x01D4171D

FIG 19(14)

1929

a=3, b=4

c/d	0	1	2	3	4	5	6	7
0	0x02103A49	0x01F00632	0x01E0052B	0x01D0041E	0x01C0070A	0x01C8041A	0x01D80522	0x01E80429
1	0x01F00532	0x01D0121E	0x01A0020C	0x01700206	0x01600209	0x01680406	0x0198030C	0x01C8111A
2	0x01E0042B	0x01A0000C	0x01500007	0x009D0001	0x00600001	0x009E0001	0x01480006	0x0198010C
3	0x01D0031E	0x01700106	0x009F0001	0x00250001	0x000A0001	0x00260001	0x00A00001	0x01680306
4	0x01BC090A	0x015C0309	0x00610001	0x000B0001	0x00010001	0x000C0001	0x00620001	0x01580208
5	0x01CC031B	0x016C0306	0x00A10001	0x00270001	0x000D0001	0x00280001	0x00A20001	0x01640206
6	0x01DC0426	0x019C010C	0x014C0506	0x00A30001	0x00630001	0x00A40001	0x01440506	0x0194060C
7	0x01EC042C	0x01CC0F1B	0x019C000C	0x016C0206	0x01540208	0x01640006	0x0194050C	0x01C41618

1930

a=3, b=5

c/d	0	1	2	3	4	5	6	7
0	0x020E202D	0x01FE2E30	0x01DE1023	0x01CE0D1B	0x01CE041B	0x01C60D17	0x01D6061A	0x01F61F22
1	0x01FE2930	0x01DE1123	0x01AE0B0C	0x017E0506	0x016E0006	0x01760406	0x01A6090C	0x01D60A1A
2	0x01DE0E23	0x01AE090C	0x015E0409	0x00FD0001	0x00A50001	0x00FE0001	0x01560708	0x01A60A0C
3	0x01CE0C1B	0x017E0306	0x00FF0001	0x00460001	0x00290001	0x00470001	0x01000001	0x01760206
4	0x01CA0318	0x016A0006	0x00A60001	0x002A0001	0x000E0001	0x002B0001	0x00A70001	0x01660106
5	0x01CA0E18	0x017A0406	0x01010001	0x00480001	0x002C0001	0x00490001	0x01020001	0x01720506
6	0x01DA0E1D	0x01AA0A0C	0x015A0509	0x01030001	0x00A80001	0x01040001	0x01520208	0x01A2050C
7	0x01FA1E24	0x01DA0A1D	0x01AA080C	0x017A0306	0x01620306	0x01720106	0x01A2080C	0x01D20F15

FIG 19(15)

1931

a=3, b=6

c/d	0	1	2	3	4	5	6	7
0	0x020E162D	0x01FE0630	0x01EE1C28	0x01DE0D23	0x01DE0723	0x01D6091A	0x01E6151E	0x01F60922
1	0x01FE0530	0x01EE1828	0x01CE031B	0x01AE070C	0x019E0B0C	0x01A6070C	0x01C60517	0x01E60E1E
2	0x01EE1B28	0x01CE021B	0x018E0507	0x015E0209	0x014E0507	0x01560108	0x01860306	0x01C60317
3	0x01DE0C23	0x01AE060C	0x015E0009	0x01050001	0x00A90001	0x01060001	0x01560008	0x01A6060C
4	0x01DA071D	0x019A0B0C	0x014A0406	0x00AA0001	0x00640001	0x00AB0001	0x01460406	0x01960B0C
5	0x01DA0C1D	0x01AA060C	0x015A0309	0x01070001	0x00AC0001	0x01080001	0x01520108	0x01A2070C
6	0x01EA1822	0x01CA0218	0x018A0107	0x015A0209	0x01420306	0x01520508	0x01820106	0x01C21115
7	0x01FA0424	0x01EA1522	0x01CA0118	0x01AA040C	0x0192040C	0x01A2020C	0x01C21015	0x01E20919

1932

a=3, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021D191A	0x020E012D	0x01FE0430	0x01FE2630	0x01EE0528	0x01F61422	0x01F61322	0x02061618
1	0x020E002D	0x01FE1430	0x01EE1628	0x01DE0623	0x01CE0B1B	0x01D60D1A	0x01E60B1E	0x01F60D22
2	0x01FE0330	0x01EE1528	0x01CE001B	0x01AE040C	0x019E0A0C	0x01A6030C	0x01C60B17	0x01E60A1E
3	0x01FE2530	0x01DE0523	0x01AE020C	0x017E0106	0x016E0506	0x01760506	0x01A6010C	0x01D6191A
4	0x01EA0522	0x01CA0818	0x019A0A0C	0x016A0506	0x015A0009	0x01660506	0x0196090C	0x01C60C17
5	0x01FA2124	0x01DA091D	0x01AA030C	0x017A0206	0x01620406	0x01720206	0x01A20B0C	0x01D20415
6	0x01FA0324	0x01EA1322	0x01CA0018	0x01AA000C	0x01920B0C	0x01A2040C	0x01C20615	0x01E20219
7	0x020A001B	0x01FA0F24	0x01EA1022	0x01DA031D	0x01C20C15	0x01D20615	0x01E20B19	0x01F20716

FIG 19(16)

1933

a=4, b=0

c/d	0	1	2	3	4	5	6	7
0	0x021E0F1D	0x021E1A1D	0x020F2732	0x020F2832	0x01FF2833	0x0207001B	0x0207041B	0x02160409
1	0x021E161D	0x020F2032	0x01FF3133	0x01EF0E2C	0x01EF212C	0x01E70822	0x01F72124	0x02070F1B
2	0x020F2532	0x01FF3033	0x01EF112C	0x01DF1B25	0x01DF0625	0x01D70A1E	0x01E70A22	0x01F71024
3	0x020F2632	0x01EF0C2C	0x01DF1825	0x01CF001B	0x01BF080A	0x01C71718	0x01D70F1E	0x01E70922
4	0x01FB2627	0x01EB1E26	0x01DB0620	0x01BB080A	0x01BF030A	0x01B70809	0x01D7051E	0x01E70622
5	0x020B141E	0x01EB0B26	0x01DB1220	0x01CB1819	0x01B30208	0x01C31417	0x01D30E18	0x01E30A1D
6	0x020B151E	0x01FB2427	0x01EB0E26	0x01DB1020	0x01D30918	0x01D31018	0x01E31C1D	0x01F30E18
7	0x021A080A	0x020B111E	0x01FB2227	0x01EB0A26	0x01E3161D	0x01E30F1D	0x01F30918	0x0203070C

1934

a=4, b=1

c/d	0	1	2	3	4	5	6	7
0	0x021E111D	0x020F1E32	0x01FF2C33	0x01EF0B2C	0x01EF202C	0x01E70222	0x01F71924	0x02070E1B
1	0x020F1D32	0x01FF2733	0x01DF2025	0x01CF141B	0x01CF1A1B	0x01C71618	0x01D7151E	0x01F71D24
2	0x01FF2B33	0x01DF1F25	0x01BF070A	0x019F0A0C	0x018F0107	0x0197090C	0x01B70609	0x01D7161E
3	0x01EF092C	0x01CF121B	0x019F080C	0x016F0406	0x015F0009	0x01670106	0x0197070C	0x01C71118
4	0x01EB1C26	0x01CB1719	0x018B0207	0x015B0009	0x014F0507	0x01570008	0x01870106	0x01C71418
5	0x01EB0926	0x01CB1319	0x019B090C	0x016B0406	0x01530508	0x01630506	0x01930B0C	0x01C31117
6	0x01FB1F27	0x01DB1A20	0x01BB070A	0x019B080C	0x01830306	0x0193040C	0x01B30308	0x01D30B18
7	0x020B0F1E	0x01FB1D27	0x01DB1920	0x01CB1119	0x01C30C17	0x01C30717	0x01D31118	0x01F30418

FIG 19(17)

1935

a=4, b=2

c/d	0	1	2	3	4	5	6	7
0	0x020F2432	0x01FF2A33	0x01EF102C	0x01DF1225	0x01DF0125	0x01D7021E	0x01E71022	0x01F72024
1	0x01FF2933	0x01DF1E25	0x01BF060A	0x019F070C	0x018F0007	0x0197060C	0x01B70509	0x01D7171E
2	0x01EF0F2C	0x01BF050A	0x017F0306	0x014F0407	0x012F0001	0x01470306	0x01770206	0x01B70409
3	0x01DF1125	0x019F060C	0x014F0207	0x00AD0001	0x00650001	0x00AE0001	0x01470506	0x01970B0C
4	0x01DB0120	0x018B0107	0x01300001	0x00660001	0x004A0001	0x00670001	0x01310001	0x01870206
5	0x01DB0E20	0x019B070C	0x014B0206	0x00AF0001	0x00680001	0x00B00001	0x01430306	0x0193090C
6	0x01EB0D26	0x01BB060A	0x017B0306	0x014B0106	0x01320001	0x01430006	0x01730206	0x01B30008
7	0x01FB2027	0x01DB1820	0x01BB050A	0x019B050C	0x01830506	0x0193050C	0x01B30508	0x01D30418

1936

a=4, b=3

c/d	0	1	2	3	4	5	6	7
0	0x020F2132	0x01EF082C	0x01DF0F25	0x01CF191B	0x01BF040A	0x01C71318	0x01D71B1E	0x01E70522
1	0x01EF072C	0x01CF111B	0x019F030C	0x016F0306	0x015F0809	0x01670206	0x0197080C	0x01C70B18
2	0x01DF0D25	0x019F020C	0x014F0007	0x00B10001	0x00690001	0x00B20001	0x01470106	0x0197020C
3	0x01CF181B	0x016F0106	0x00B30001	0x002D0001	0x000F0001	0x002E0001	0x00B40001	0x01670306
4	0x01BB040A	0x015B0809	0x006A0001	0x00100001	0x00020001	0x00110001	0x006B0001	0x01570508
5	0x01CB1619	0x016B0206	0x00B50001	0x002F0001	0x00120001	0x00300001	0x00B60001	0x01630206
6	0x01DB0B20	0x019B030C	0x014B0006	0x00B70001	0x006C0001	0x00B80001	0x01430206	0x01930A0C
7	0x01EB0726	0x01CB0F19	0x019B020C	0x016B0106	0x01530208	0x01630106	0x0193030C	0x01C31517

FIG 19(18)

1937

a=4, b=4

c/d	0	1	2	3	4	5	6	7
0	0x01FD2025	0x01ED1A24	0x01DD1D1F	0x01BD070B	0x01BD050B	0x01B50508	0x01D50E16	0x01E50A19
1	0x01ED1924	0x01CD1619	0x018D0309	0x015D080B	0x014D0108	0x0155050A	0x01850108	0x01C51216
2	0x01DD191F	0x018D0009	0x01330001	0x006D0001	0x004B0001	0x006E0001	0x01340001	0x01850008
3	0x01BD060B	0x015D060B	0x006F0001	0x00130001	0x00030001	0x00140001	0x00700001	0x0155070A
4	0x01BB030A	0x01490007	0x004C0001	0x00040001	0x00000001	0x00050001	0x004D0001	0x01450107
5	0x01B90509	0x0159050B	0x00710001	0x00150001	0x00060001	0x00160001	0x00720001	0x0151050A
6	0x01D91419	0x01890109	0x01350001	0x00730001	0x004E0001	0x00740001	0x01360001	0x01810508
7	0x01E9111D	0x01C91117	0x01890009	0x0159040B	0x01410007	0x0151010A	0x01810008	0x01C10915

1938

a=4, b=5

c/d	0	1	2	3	4	5	6	7
0	0x020E1F2D	0x01EE0428	0x01DE0323	0x01CE161B	0x01BE0709	0x01C61117	0x01D6111A	0x01E6001E
1	0x01EE0328	0x01CE0A1B	0x019E050C	0x016E0306	0x015E0709	0x01660306	0x0196040C	0x01C60417
2	0x01DE0223	0x019E040C	0x014E0207	0x00B90001	0x00750001	0x00BA0001	0x01460206	0x0196030C
3	0x01CE131B	0x016E0206	0x00BB0001	0x00310001	0x00170001	0x00320001	0x00BC0001	0x01660206
4	0x01BA0508	0x015A0809	0x00760001	0x00180001	0x00070001	0x00190001	0x00770001	0x01560508
5	0x01CA1218	0x016A0306	0x00BD0001	0x00330001	0x001A0001	0x00340001	0x00BE0001	0x01620106
6	0x01DA051D	0x019A050C	0x014A0106	0x00BF0001	0x00780001	0x00C00001	0x01420206	0x0192060C
7	0x01EA0322	0x01CA0918	0x019A040C	0x016A0206	0x01520408	0x01620506	0x0192000C	0x01C21415

FIG 19(19)

1939

a=4, b=6

c/d	0	1	2	3	4	5	6	7
0	0x020E212D	0x01FE2430	0x01EE0728	0x01DE0123	0x01DE2223	0x01D6081A	0x01E6161E	0x01F60C22
1	0x01FE2330	0x01DE0B23	0x01BE0609	0x019E030C	0x018E0007	0x0196020C	0x01B60507	0x01D6151A
2	0x01EE0628	0x01BE0509	0x017E0006	0x014E0107	0x01370001	0x01460106	0x01760006	0x01B60307
3	0x01DE0023	0x019E020C	0x014E0007	0x00C10001	0x00790001	0x00C20001	0x01460006	0x0196010C
4	0x01DA1C1D	0x018A0007	0x01380001	0x007A0001	0x004F0001	0x007B0001	0x01390001	0x01860006
5	0x01DA041D	0x019A030C	0x014A0006	0x00C30001	0x007C0001	0x00C40001	0x01420006	0x0192030C
6	0x01EA2022	0x01BA0408	0x017A0106	0x014A0506	0x013A0001	0x01420506	0x01720006	0x01B20507
7	0x01FA1C24	0x01DA081D	0x01BA0308	0x019A020C	0x01820406	0x0192080C	0x01B20407	0x01D20A15

1940

a=4, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021D011A	0x020E1E2D	0x01FE2230	0x01EE0228	0x01EE1A28	0x01E61D1E	0x01F61D22	0x02061718
1	0x020E1C2D	0x01FE1E30	0x01DE0923	0x01CE071B	0x01CE111B	0x01C60617	0x01D6071A	0x01F61922
2	0x01FE2130	0x01DE0823	0x01BE0309	0x019E010C	0x018E0607	0x0196000C	0x01B60407	0x01D6011A
3	0x01EE0128	0x01CE061B	0x019E000C	0x016E0106	0x015E0609	0x01660006	0x01960A0C	0x01C60117
4	0x01EA1622	0x01CA1018	0x018A0507	0x015A0709	0x014E0607	0x01560308	0x01860506	0x01C60917
5	0x01EA0222	0x01CA0718	0x019A010C	0x016A0106	0x01520708	0x01620206	0x0192010C	0x01C21315
6	0x01FA1B24	0x01DA001D	0x01BA0208	0x019A000C	0x01820206	0x0192070C	0x01B20207	0x01D20515
7	0x020A171B	0x01FA1624	0x01DA011D	0x01CA0418	0x01C20815	0x01C20915	0x01D20C15	0x01F20116

FIG 19(20)

1941

a=5, b=0

c/d	0	1	2	3	4	5	6	7
0	0x021E0A1D	0x021E141D	0x020F1632	0x020F1A32	0x020F1F32	0x02070A1B	0x0207111B	0x02160309
1	0x021E121D	0x020F0032	0x01FF1333	0x01FF2233	0x01EF062C	0x01F71A24	0x01F71224	0x02070C1B
2	0x020F1532	0x01FF1233	0x01EF1E2C	0x01DF1C25	0x01DF0925	0x01D7081E	0x01E71522	0x01F71324
3	0x020F1932	0x01FF2133	0x01DF1A25	0x01CF0F1B	0x01CF171B	0x01C70918	0x01D70B1E	0x01F71824
4	0x020B101E	0x01EB0626	0x01DB0920	0x01CB1519	0x01B70309	0x01C70A18	0x01D7121E	0x01E70322
5	0x020B011E	0x01FB1927	0x01DB1420	0x01CB0D19	0x01C30917	0x01C30017	0x01D30F18	0x01F30118
6	0x020B0B1E	0x01FB0B27	0x01EB1A26	0x01DB1320	0x01D31518	0x01D30A18	0x01E30C1D	0x01F31318
7	0x021A030A	0x020B001E	0x01FB0A27	0x01FB1727	0x01E31B1D	0x01F30018	0x01F30B18	0x0203090C

1942

a=5. b=1

c/d	0	1	2	3	4	5	6	7
0	0x021E101D	0x020F3132	0x01FF1133	0x01FF2033	0x01EF042C	0x01F71424	0x01F70524	0x02071A1B
1	0x020F3032	0x01FF1F33	0x01EF142C	0x01DF2325	0x01CF0E1B	0x01D71C1E	0x01E70C22	0x01F71524
2	0x01FF1033	0x01EF132C	0x01CF161B	0x01AF050C	0x019F010C	0x01A7060C	0x01C70F18	0x01E70B22
3	0x01FF1E33	0x01DF2225	0x01AF040C	0x017F0206	0x016F0006	0x01770306	0x01A7050C	0x01D71A1E
4	0x01EB0526	0x01CB0C19	0x019B010C	0x016B0006	0x015F0709	0x01670006	0x0197010C	0x01C70718
5	0x01FB1527	0x01DB1E20	0x01AB050C	0x017B0206	0x01630306	0x01730006	0x01A3090C	0x01D30518
6	0x01FB0927	0x01EB1126	0x01CB1419	0x01AB040C	0x0193010C	0x01A3030C	0x01C30417	0x01E3101D
7	0x020B1D1E	0x01FB1627	0x01EB1026	0x01DB1D20	0x01C31317	0x01D31718	0x01E3081D	0x01F30A18

FIG 19(21)

1943

a=5, b=2

c/d	0	1	2	3	4	5	6	7
0	0x020F1432	0x01FF0E33	0x01EF1C2C	0x01DF1725	0x01DF0725	0x01D70E1E	0x01E71222	0x01F71B24
1	0x01FF0D33	0x01EF122C	0x01CF151B	0x01AF030C	0x019F000C	0x01A7080C	0x01C70D18	0x01E70422
2	0x01EF1B2C	0x01CF131B	0x018F0607	0x015F0609	0x014F0607	0x01570708	0x01870506	0x01C71018
3	0x01DF1625	0x01AF020C	0x015F0509	0x01090001	0x00C50001	0x010A0001	0x01570608	0x01A7020C
4	0x01DB0720	0x019B000C	0x014B0506	0x00C60001	0x007D0001	0x00C70001	0x01470406	0x0197000C
5	0x01DB1120	0x01AB030C	0x015B0509	0x010B0001	0x00C80001	0x010C0001	0x01530408	0x01A3020C
6	0x01EB1926	0x01CB1219	0x018B0607	0x015B0409	0x01430106	0x01530008	0x01830206	0x01C31017
7	0x01FB0827	0x01EB0F26	0x01CB1019	0x01AB020C	0x0193080C	0x01A3000C	0x01C30F17	0x01E3111D

1944

a=5, b=3

c/d	0	1	2	3	4	5	6	7
0	0x020F1832	0x01FF1D33	0x01DF1425	0x01CF0D1B	0x01CF101B	0x01C70818	0x01D70C1E	0x01F71124
1	0x01FF1B33	0x01DF2125	0x01AF010C	0x017F0106	0x016F0506	0x01770106	0x01A7010C	0x01D7191E
2	0x01DF1325	0x01AF000C	0x015F0409	0x010D0001	0x00C90001	0x010E0001	0x01570408	0x01A7000C
3	0x01CF0C1B	0x017F0006	0x010F0001	0x00500001	0x00350001	0x00510001	0x01100001	0x01770006
4	0x01CB0E19	0x016B0506	0x00CA0001	0x00360001	0x001B0001	0x00370001	0x00CB0001	0x01670406
5	0x01CB0B19	0x017B0106	0x01110001	0x00520001	0x00380001	0x00530001	0x01120001	0x01730506
6	0x01DB1620	0x01AB010C	0x015B0309	0x01130001	0x00CC0001	0x01140001	0x01530608	0x01A3040C
7	0x01FB1227	0x01DB1B20	0x01AB000C	0x017B0006	0x01630006	0x01730406	0x01A3080C	0x01D30318

FIG 19(22)

1945

a=5, b=4

c/d	0	1	2	3	4	5	6	7
0	0x020D101B	0x01ED0524	0x01DD061F	0x01CD1519	0x01BD040B	0x01C50F16	0x01D50116	0x01E51019
1	0x01ED0424	0x01CD1019	0x019D0B0C	0x016D0506	0x015D050B	0x01650506	0x01950B0C	0x01C50A16
2	0x01DD041F	0x019D0A0C	0x014D0708	0x00CD0001	0x007E0001	0x00CE0001	0x01450407	0x0195090C
3	0x01CD1419	0x016D0406	0x00CF0001	0x00390001	0x001C0001	0x003A0001	0x00D00001	0x01650306
4	0x01B90409	0x0159030B	0x007F0001	0x001D0001	0x00080001	0x001E0001	0x00800001	0x0155010A
5	0x01C91017	0x01690506	0x00D10001	0x003B0001	0x001F0001	0x003C0001	0x00D20001	0x01610306
6	0x01D91119	0x01990B0C	0x01490607	0x00D30001	0x00810001	0x00D40001	0x01410307	0x0191020C
7	0x01E90F1D	0x01C91617	0x01990A0C	0x01690406	0x0151040A	0x01610006	0x0191030C	0x01C10D15

1946

a=5, b=5

c/d	0	1	2	3	4	5	6	7
0	0x020D0E1B	0x01FD1725	0x01DD121F	0x01CD0F19	0x01CD1319	0x01C50C16	0x01D50316	0x01F50216
1	0x01FD1525	0x01DD161F	0x01AD010C	0x017D0506	0x016D0306	0x01750506	0x01A5010C	0x01D50B16
2	0x01DD111F	0x01AD000C	0x015D040B	0x01150001	0x00D50001	0x01160001	0x0155040A	0x01A5000C
3	0x01CD0E19	0x017D0406	0x01170001	0x00540001	0x003D0001	0x00550001	0x01180001	0x01750106
4	0x01C90C17	0x01690306	0x00D60001	0x003E0001	0x00200001	0x003F0001	0x00D70001	0x01650406
5	0x01C90D17	0x01790506	0x01190001	0x00560001	0x00400001	0x00570001	0x011A0001	0x01710406
6	0x01D90419	0x01A9010C	0x0159020B	0x011B0001	0x00D80001	0x011C0001	0x0151090A	0x01A1030C
7	0x01F90918	0x01D90C19	0x01A9000C	0x01790306	0x01610506	0x01710106	0x01A1040C	0x01D10F10

FIG 19(23)

1947

a=5, b=6

c/d	0	1	2	3	4	5	6	7
0	0x020D0D1B	0x01FD0F25	0x01ED1824	0x01DD101F	0x01DD011F	0x01D50416	0x01E50419	0x01F50E16
1	0x01FD0E25	0x01ED0B24	0x01CD1219	0x01AD0B0C	0x019D090C	0x01A5050C	0x01C50816	0x01E50519
2	0x01ED1724	0x01CD1119	0x018D0809	0x015D030B	0x014D0608	0x0155080A	0x01850308	0x01C51516
3	0x01DD0F1F	0x01AD0A0C	0x015D020B	0x011D0001	0x00D90001	0x011E0001	0x0155000A	0x01A50B0C
4	0x01D90119	0x0199090C	0x01490507	0x00DA0001	0x00820001	0x00DB0001	0x01450607	0x0195020C
5	0x01D90519	0x01A90B0C	0x0159010B	0x011F0001	0x00DC0001	0x01200001	0x0151000A	0x01A1010C
6	0x01E90E1D	0x01C90F17	0x01890809	0x0159000B	0x01410207	0x0151030A	0x01810708	0x01C10615
7	0x01F91318	0x01E9021D	0x01C90E17	0x01A90A0C	0x01910B0C	0x01A1020C	0x01C10315	0x01E10612

1948

a=5, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021C0209	0x020D191B	0x01FD0D25	0x01FD1325	0x01ED0324	0x01F50C16	0x01F50816	0x0205040B
1	0x020D181B	0x01FD1825	0x01ED0924	0x01DD151F	0x01CD0D19	0x01D50916	0x01E50219	0x01F50F16
2	0x01FD0C25	0x01ED0824	0x01CD0C19	0x01AD090C	0x019D080C	0x01A50A0C	0x01C50416	0x01E50319
3	0x01FD1225	0x01DD141F	0x01AD080C	0x017D0306	0x016D0206	0x01750206	0x01A5090C	0x01D50816
4	0x01E9101D	0x01C90217	0x0199080C	0x01690206	0x015D010B	0x01650106	0x0195080C	0x01C50616
5	0x01F90718	0x01D90B19	0x01A9090C	0x01790206	0x01610206	0x01710506	0x01A10B0C	0x01D10C10
6	0x01F90518	0x01E9011D	0x01C90817	0x01A9080C	0x0191000C	0x01A1000C	0x01C10F15	0x01E10112
7	0x02090B0C	0x01F90B18	0x01E9041D	0x01D90819	0x01C10415	0x01D10310	0x01E10912	0x01F1050A

FIG 19(24)

1949

a=6, b=0

c/d	0	1	2	3	4	5	6	7
0	0x021E051D	0x021E191D	0x021E1B1D	0x020F1232	0x020F2332	0x02070D1B	0x02160109	0x02160209
1	0x021E181D	0x020F0B32	0x020F1C32	0x01FF0B33	0x01FF2633	0x01F70224	0x0207021B	0x0207081B
2	0x021E171D	0x020F1B32	0x01FF1C33	0x01EF1A2C	0x01EF0D2C	0x01E71622	0x01F72224	0x0207191B
3	0x020F1032	0x01FF0A33	0x01EF192C	0x01DF1025	0x01DF0525	0x01D7031E	0x01E70F22	0x01F71724
4	0x020B131E	0x01FB1E27	0x01EB0C26	0x01DB0520	0x01D7041E	0x01D7111E	0x01E70022	0x01F71C24
5	0x020B0A1E	0x01FB0627	0x01EB1626	0x01DB0D20	0x01D30218	0x01D31418	0x01E3181D	0x01F30318
6	0x021A040A	0x020B0E1E	0x01FB1327	0x01EB1526	0x01E3051D	0x01E3061D	0x01F31518	0x0203080C
7	0x021A090A	0x020B021E	0x020B0D1E	0x01FB0527	0x01F31118	0x01F31418	0x0203060C	0x0203020C

1950

a=6, b=1

c/d	0	1	2	3	4	5	6	7
0	0x021E151D	0x020F0932	0x020F1132	0x01FF0933	0x01FF2533	0x01F70124	0x02070B1B	0x0207101B
1	0x020F0832	0x01FF3233	0x01EF022C	0x01EF052C	0x01DF0E25	0x01E71422	0x01E71C22	0x01F70A24
2	0x020F0E32	0x01EF012C	0x01DF0425	0x01CF091B	0x01BF020A	0x01C70C18	0x01D7061E	0x01E70122
3	0x01FF0833	0x01EF032C	0x01CF081B	0x01AF0B0C	0x019F0B0C	0x01A70B0C	0x01C71518	0x01E71922
4	0x01FB1C27	0x01DB1C20	0x01BB020A	0x019B0B0C	0x018F0507	0x01970A0C	0x01B70209	0x01D7071E
5	0x01FB0327	0x01EB0226	0x01CB0819	0x01AB0B0C	0x0193020C	0x01A30B0C	0x01C30517	0x01E3151D
6	0x020B071E	0x01EB0426	0x01DB0420	0x01CB0719	0x01B30708	0x01C30B17	0x01D30118	0x01E30E1D
7	0x020B081E	0x01FB2527	0x01EB0326	0x01EB0026	0x01D30018	0x01E3091D	0x01E31A1D	0x01F30818

FIG 19(25)

1951

a=6, b=2

c/d	0	1	2	3	4	5	6	7
0	0x021E0C1D	0x020F0D32	0x01FF1A33	0x01EF182C	0x01EF0A2C	0x01E70E22	0x01F70D24	0x0207031B
1	0x020F0C32	0x01EF002C	0x01DF0325	0x01CF071B	0x01BF000A	0x01C70418	0x01D7141E	0x01E71D22
2	0x01FF1933	0x01DF0225	0x01BF010A	0x018F0407	0x017F0506	0x01870406	0x01B70009	0x01D7011E
3	0x01EF172C	0x01CF061B	0x018F0307	0x015F0309	0x014F0307	0x01570108	0x01870306	0x01C70318
4	0x01EB0826	0x01BB010A	0x017B0506	0x014B0406	0x013B0001	0x01470206	0x01770506	0x01B70109
5	0x01EB1426	0x01CB0619	0x018B0507	0x015B0209	0x01430406	0x01530708	0x01830406	0x01C30A17
6	0x01FB1127	0x01DB0320	0x01BB000A	0x018B0407	0x01730306	0x01830006	0x01B30408	0x01D30C18
7	0x020B061E	0x01EB0126	0x01DB0220	0x01CB0519	0x01B30608	0x01C30817	0x01D31318	0x01E30B1D

1952

a=6, b=3

c/d	0	1	2	3	4	5	6	7
0	0x020F0F32	0x01FF0733	0x01EF162C	0x01DF0C25	0x01DF0025	0x01D7001E	0x01E70722	0x01F70024
1	0x01FF0633	0x01EF2B2C	0x01CF051B	0x01AF0A0C	0x019F090C	0x01A70A0C	0x01C71218	0x01E72022
2	0x01EF152C	0x01CF041B	0x018F0207	0x015F0209	0x014F0107	0x01570308	0x01870006	0x01C70218
3	0x01DF0B25	0x01AF090C	0x015F0109	0x01210001	0x00DD0001	0x01220001	0x01570208	0x01A7090C
4	0x01DB0020	0x019B0A0C	0x014B0306	0x00DE0001	0x00830001	0x00DF0001	0x01470006	0x0197040C
5	0x01DB0F20	0x01AB0A0C	0x015B0109	0x01230001	0x00E00001	0x01240001	0x01530108	0x01A3060C
6	0x01EB1326	0x01CB0419	0x018B0307	0x015B0609	0x01430506	0x01530308	0x01830106	0x01C31217
7	0x01FB0427	0x01EB2326	0x01CB0319	0x01AB090C	0x0193070C	0x01A3070C	0x01C30D17	0x01E30D1D

FIG 19(26)

1953

a=6, b=4

c/d	0	1	2	3	4	5	6	7
0	0x020D121B	0x01FD1F25	0x01ED0724	0x01DD001F	0x01DD131F	0x01D51516	0x01E51519	0x01F51416
1	0x01FD1E25	0x01DD0E1F	0x01BD030B	0x019D070C	0x018D0709	0x0195050C	0x01B50208	0x01D50F16
2	0x01ED0624	0x01BD020B	0x017D0206	0x014D0508	0x013C0001	0x01450207	0x01750306	0x01B50608
3	0x01DD1E1F	0x019D060C	0x014D0408	0x00E10001	0x00840001	0x00E20001	0x01450507	0x01950A0C
4	0x01D90619	0x01890709	0x013D0001	0x00850001	0x00580001	0x00860001	0x013E0001	0x01850708
5	0x01D91819	0x0199070C	0x01490407	0x00E30001	0x00870001	0x00E40001	0x01410607	0x01910A0C
6	0x01E9031D	0x01B90309	0x01790106	0x01490307	0x013F0001	0x01410507	0x01710006	0x01B10007
7	0x01F91118	0x01D90719	0x01B90209	0x0199010C	0x01810108	0x0191060C	0x01B10407	0x01D10A10

1954

a=6, b=5

c/d	0	1	2	3	4	5	6	7
0	0x020D0C1B	0x01FD0A25	0x01ED1424	0x01DD0C1F	0x01DD1C1F	0x01D51316	0x01E50719	0x01F50716
1	0x01FD0925	0x01ED2324	0x01CD0719	0x01AD070C	0x019D050C	0x01A5020C	0x01C50516	0x01E50F19
2	0x01ED1324	0x01CD0619	0x018D0609	0x015D000B	0x014D0308	0x0155030A	0x01850508	0x01C51416
3	0x01DD0B1F	0x01AD060C	0x015D0A0B	0x01250001	0x00E50001	0x01260001	0x0155090A	0x01A5070C
4	0x01D91719	0x0199060C	0x01490207	0x00E60001	0x00880001	0x00E70001	0x01450307	0x0195060C
5	0x01D90919	0x01A9070C	0x01590A0B	0x01270001	0x00E80001	0x01280001	0x0151060A	0x01A1060C
6	0x01E90B1D	0x01C90917	0x01890609	0x0159090B	0x01410407	0x0151020A	0x01810308	0x01C11015
7	0x01F90418	0x01E9191D	0x01C90717	0x01A9060C	0x0191090C	0x01A1050C	0x01C10E15	0x01E10A12

FIG 19(27)

1955

a=6, b=6

c/d	0	1	2	3	4	5	6	7
0	0x021C0509	0x020D0A1B	0x01FD1625	0x01ED1124	0x01ED0A24	0x01E50E19	0x01F50316	0x0205010B
1	0x020D091B	0x01ED0224	0x01DD1B1F	0x01CD0519	0x01BD010B	0x01C50716	0x01D50016	0x01E50919
2	0x01FD1425	0x01DD1A1F	0x01BD080B	0x018D0509	0x017D0106	0x01850208	0x01B50008	0x01D50716
3	0x01ED1024	0x01CD0419	0x018D0409	0x015D090B	0x014D0208	0x0155020A	0x01850408	0x01C50316
4	0x01E91A1D	0x01B90109	0x01790006	0x01490107	0x01400001	0x01450007	0x01750406	0x01B50308
5	0x01E90A1D	0x01C90617	0x01890509	0x0159080B	0x01410107	0x0151080A	0x01810608	0x01C10715
6	0x01F90818	0x01D91619	0x01B90609	0x01890409	0x01710306	0x01810208	0x01B10207	0x01D10510
7	0x0209030C	0x01E9001D	0x01D91519	0x01C90517	0x01B10307	0x01C10515	0x01D10D10	0x01E11012

1956

a=6, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021C0409	0x020D061B	0x020D081B	0x01FD0725	0x01FD1D25	0x01F51016	0x0205080B	0x02050A0B
1	0x020D051B	0x01FD2425	0x01ED0124	0x01ED1D24	0x01DD0A1F	0x01E51419	0x01E51719	0x01F51116
2	0x020D071B	0x01ED0024	0x01DD181F	0x01CD0319	0x01BD000B	0x01C50216	0x01D50616	0x01E51619
3	0x01FD0625	0x01ED1B24	0x01CD0219	0x01AD050C	0x019D040C	0x01A5060C	0x01C50116	0x01E50D19
4	0x01F91018	0x01D90A19	0x01B90009	0x0199050C	0x018D0209	0x0195040C	0x01B50408	0x01D51216
5	0x01F90318	0x01E9141D	0x01C90417	0x01A9050C	0x0191070C	0x01A10A0C	0x01C10A15	0x01E10412
6	0x0209060C	0x01E91C1D	0x01D91319	0x01C90317	0x01B10507	0x01C11415	0x01D10710	0x01E10812
7	0x0209000C	0x01F91518	0x01E91B1D	0x01E9131D	0x01D10810	0x01E10312	0x01E10B12	0x01F1060A

FIG 19(28)

1957

a=7, b=0

c/d	0	1	2	3	4	5	6	7
0	0x021E1C1D	0x021E081D	0x021E041D	0x021E0E1D	0x021E0D1D	0x02160709	0x02160009	0x02160809
1	0x021E071D	0x021E031D	0x020F0732	0x020F2F32	0x020F1732	0x0207151B	0x0207071B	0x02160509
2	0x021E131D	0x020F0632	0x020F0A32	0x01FF0533	0x01FF2433	0x01F70C24	0x0207061B	0x0207121B
3	0x021E0B1D	0x020F2E32	0x01FF0433	0x01FF1533	0x01EF2A2C	0x01F70724	0x01F70324	0x0207131B
4	0x021A050A	0x020B0C1E	0x01FB1B27	0x01EB2526	0x01E3041D	0x01E72122	0x01F70824	0x0207171B
5	0x021A060A	0x020B1C1E	0x01FB0227	0x01FB0D27	0x01E3001D	0x01F30518	0x01F31618	0x0203040C
6	0x021A000A	0x020B041E	0x020B051E	0x01FB0127	0x01F30618	0x01F30218	0x0203050C	0x0203010C
7	0x021A070A	0x021A020A	0x020B031E	0x020B1B1E	0x02030A0C	0x0203030C	0x02030B0C	0x02120001

1958

a=7, b=1

c/d	0	1	2	3	4	5	6	7
0	0x021E061D	0x021E011D	0x020F0432	0x020F2D32	0x020F1332	0x0207141B	0x0207011B	0x02160609
1	0x021E001D	0x020F2232	0x01FF2F33	0x01FF1833	0x01FF1433	0x01F70E24	0x01F70B24	0x0207091B
2	0x020F0332	0x01FF2E33	0x01EF292C	0x01EF252C	0x01DF0A25	0x01E71B22	0x01E71F22	0x01F72324
3	0x020F2C32	0x01FF1733	0x01EF242C	0x01DF1D25	0x01CF0B1B	0x01D7091E	0x01E71A22	0x01F70924
4	0x020B091E	0x01FB0C27	0x01DB0A20	0x01CB0A19	0x01C31617	0x01C70E18	0x01D7181E	0x01F70624
5	0x020B1A1E	0x01FB1027	0x01EB2026	0x01DB1720	0x01C30117	0x01D30718	0x01E3191D	0x01F31718
6	0x020B171E	0x01FB2327	0x01EB2426	0x01EB1F26	0x01D30618	0x01E3011D	0x01E3021D	0x01F30C18
7	0x021A010A	0x020B121E	0x01FB1827	0x01FB0F27	0x01F31018	0x01F30F18	0x01F30D18	0x0203000C

FIG 19(29)

1959

a=7, b=2

c/d	0	1	2	3	4	5	6	7
0	0x021E091D	0x020F2B32	0x01FF0133	0x01FF0F33	0x01EF262C	0x01F70424	0x01F70F24	0x0207181B
1	0x020F2A32	0x01FF1633	0x01EF1F2C	0x01DF1925	0x01CF0A1B	0x01D7101E	0x01E71322	0x01F71E24
2	0x01FF0033	0x01EF1D2C	0x01CF011B	0x01AF070C	0x019F040C	0x01A7030C	0x01C70618	0x01E71122
3	0x01FF0C33	0x01DF1525	0x01AF060C	0x017F0406	0x016F0206	0x01770406	0x01A7070C	0x01D7131E
4	0x01EB1226	0x01CB0919	0x019B040C	0x016B0306	0x015B0709	0x01670506	0x0197030C	0x01C70518
5	0x01FB0727	0x01DB1520	0x01AB070C	0x017B0406	0x01630406	0x01730106	0x01A3050C	0x01D31218
6	0x01FB0027	0x01EB1826	0x01CB0019	0x01AB060C	0x0193060C	0x01A30A0C	0x01C30217	0x01E3071D
7	0x020B191E	0x01FB0E27	0x01EB1726	0x01DB0C20	0x01C30E17	0x01D31618	0x01E3121D	0x01F30718

1960

a=7, b=3

c/d	0	1	2	3	4	5	6	7
0	0x021E091D	0x020F2B32	0x01FF0133	0x01FF0F33	0x01EF262C	0x01F70424	0x01F70F24	0x0207181B
1	0x020F2A32	0x01FF1633	0x01EF1F2C	0x01DF1925	0x01CF0A1B	0x01D7101E	0x01E71322	0x01F71E24
2	0x01FF0033	0x01EF1D2C	0x01CF011B	0x01AF070C	0x019F040C	0x01A7030C	0x01C70618	0x01E71122
3	0x01FF0C33	0x01DF1525	0x01AF060C	0x017F0406	0x016F0206	0x01770406	0x01A7070C	0x01D7131E
4	0x01EB1226	0x01CB0919	0x019B040C	0x016B0306	0x015B0709	0x01670506	0x0197030C	0x01C70518
5	0x01FB0727	0x01DB1520	0x01AB070C	0x017B0406	0x01630406	0x01730106	0x01A3050C	0x01D31218
6	0x01FB0027	0x01EB1826	0x01CB0019	0x01AB060C	0x0193060C	0x01A30A0C	0x01C30217	0x01E3071D
7	0x020B191E	0x01FB0E27	0x01EB1726	0x01DB0C20	0x01C30E17	0x01D31618	0x01E3121D	0x01F30718

FIG 19(30)

1961

a=7, b=4

c/d	0	1	2	3	4	5	6	7
0	0x021C0709	0x020D0F1B	0x01FD1C25	0x01ED2224	0x01ED0E24	0x01E50C19	0x01F50B16	0x0205090B
1	0x020D131B	0x01FD0B25	0x01DD081F	0x01CD0B19	0x01CD0119	0x01C51116	0x01D51116	0x01F50916
2	0x01FD1B25	0x01DD071F	0x01BD0A0B	0x019D030C	0x018D0109	0x0195000C	0x01B50108	0x01D50D16
3	0x01ED2124	0x01CD0A19	0x019D020C	0x016D0106	0x015D070B	0x01650006	0x0195070C	0x01C51016
4	0x01E9091D	0x01C90117	0x01890309	0x0159070B	0x014D0008	0x0155060A	0x01850608	0x01C50916
5	0x01E9071D	0x01C90B17	0x0199040C	0x01690006	0x0151070A	0x01610106	0x0191050C	0x01C10C15
6	0x01F90A18	0x01D90E19	0x01B90709	0x0199000C	0x01810408	0x0191010C	0x01B10607	0x01D10110
7	0x0209090C	0x01F90118	0x01D90D19	0x01C91317	0x01C10115	0x01C11215	0x01D10210	0x01F1020A

1962

a=7, b=5

c/d	0	1	2	3	4	5	6	7
0	0x021C0609	0x020D171B	0x01FD0325	0x01FD0825	0x01ED2024	0x01F50616	0x01F51316	0x0205000B
1	0x020D161B	0x01FD1125	0x01ED1624	0x01DD0D1F	0x01CD0919	0x01D51416	0x01E51119	0x01F50516
2	0x01FD0225	0x01ED1524	0x01CD0019	0x01AD040C	0x019D010C	0x01A5040C	0x01C50B16	0x01E50019
3	0x01FD0525	0x01DD091F	0x01AD030C	0x017D0006	0x016D0006	0x01750006	0x01A5030C	0x01D50A16
4	0x01E9121D	0x01C90A17	0x0199020C	0x01690106	0x0159060B	0x01650206	0x0195010C	0x01C50D16
5	0x01F90218	0x01D90319	0x01A9030C	0x01790406	0x01610406	0x01710206	0x01A1090C	0x01D10410
6	0x01F90F18	0x01E90D1D	0x01C90017	0x01A9020C	0x0191040C	0x01A1070C	0x01C10015	0x01E10712
7	0x0209080C	0x01F90618	0x01E90C1D	0x01D90219	0x01C10815	0x01D10910	0x01E10C12	0x01F1010A

FIG 19(31)

1963

a=7, b=6

c/d	0	1	2	3	4	5	6	7
0	0x021C0809	0x020D041B	0x020D011B	0x01FD0125	0x01FD1A25	0x01F51516	0x0205020B	0x0205070B
1	0x020D031B	0x01FD2325	0x01ED1F24	0x01ED1224	0x01DD051F	0x01E50819	0x01E50119	0x01F50D16
2	0x020D001B	0x01ED1E24	0x01DD171F	0x01CD1819	0x01BD090B	0x01C51316	0x01D50216	0x01E51819
3	0x01FD0025	0x01ED0F24	0x01CD1719	0x01AD020C	0x019D000C	0x01A5080C	0x01C50016	0x01E50619
4	0x01F90E18	0x01D90F19	0x01B90809	0x0199030C	0x01890209	0x0195030C	0x01B50708	0x01D51016
5	0x01F91718	0x01E9081D	0x01C91517	0x01A9040C	0x0191080C	0x01A1080C	0x01C10B15	0x01E10D12
6	0x0209020C	0x01E9171D	0x01D91219	0x01C91417	0x01B10107	0x01C10215	0x01D10E10	0x01E10512
7	0x0209070C	0x01F91618	0x01E9161D	0x01E9061D	0x01D10010	0x01E11112	0x01E10F12	0x01F1090A

1964

a=7, b=7

c/d	0	1	2	3	4	5	6	7
0	0x021C0309	0x021C0109	0x020D021B	0x020D151B	0x020D0B1B	0x0205050B	0x0205060B	0x02140001
1	0x021C0009	0x020D111B	0x01FD2225	0x01FD1025	0x01FD0425	0x01F50416	0x01F50016	0x0205030B
2	0x020D1A1B	0x01FD2125	0x01ED1C24	0x01ED0D24	0x01DD021F	0x01E51219	0x01E51319	0x01F51216
3	0x020D141B	0x01FD1925	0x01ED0C24	0x01DD031F	0x01CD0819	0x01D50516	0x01E50B19	0x01F50A16
4	0x0209040C	0x01F91218	0x01D91019	0x01C91217	0x01C11315	0x01C50E16	0x01D50C16	0x01F50116
5	0x0209010C	0x01F90C18	0x01E9051D	0x01D90019	0x01C11115	0x01D10610	0x01E10212	0x01F1030A
6	0x02090A0C	0x01F91418	0x01E9151D	0x01E9181D	0x01D10B10	0x01E10012	0x01E10E12	0x01F1000A
7	0x02180001	0x0209050C	0x01F90018	0x01F90D18	0x01F1040A	0x01F1080A	0x01F1070A	0x02010001

FIG 19(32)

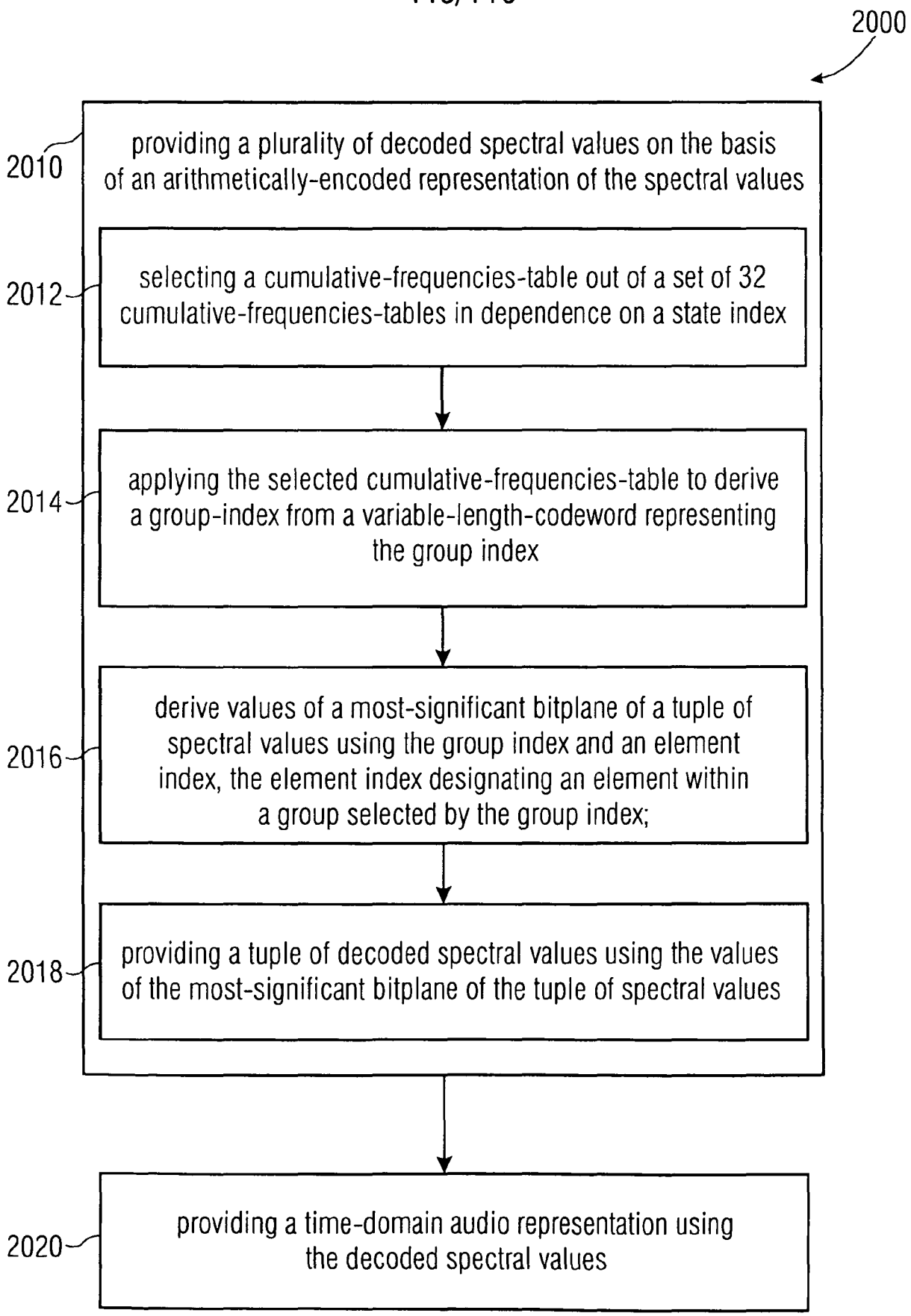

2000
2010
providing a plurality of decoded spectral values on the basis of an arithmetically-encoded representation of the spectral values
2012
selecting a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on a state index
2014
applying the selected cumulative-frequencies-table to derive a group-index from a variable-length-codeword representing the group index
2016
derive values of a most-significant bitplane of a tuple of spectral values using the group index and an element index, the element index designating an element within a group selected by the group index;
2018
providing a tuple of decoded spectral values using the values of the most-significant bitplane of the tuple of spectral values
2020
providing a time-domain audio representation using the decoded spectral values

FIG 20

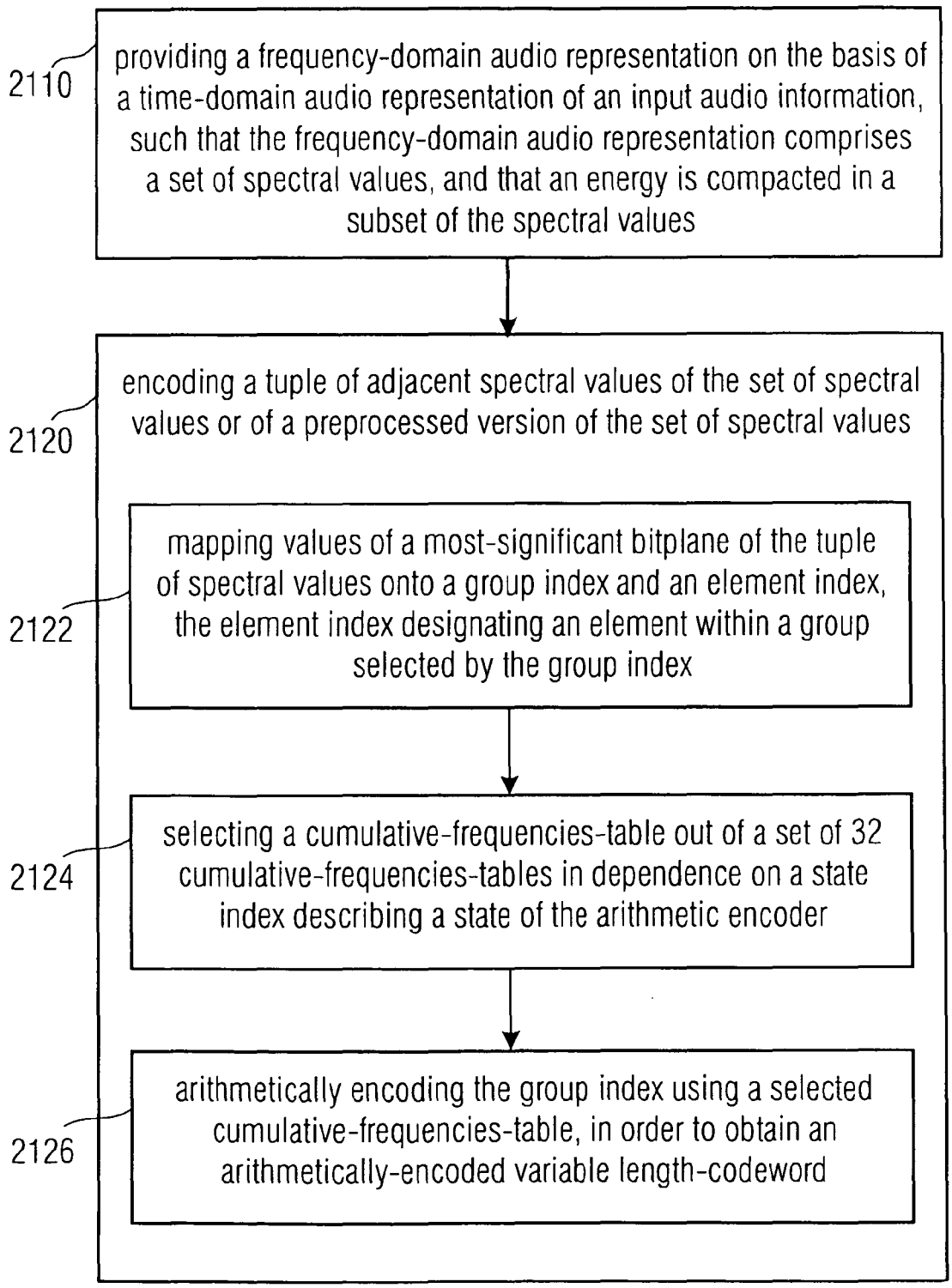

2110
providing a frequency-domain audio representation on the basis of a time-domain audio representation of an input audio information, such that the frequency-domain audio representation comprises a set of spectral values, and that an energy is compacted in a subset of the spectral values
2120
encoding a tuple of adjacent spectral values of the set of spectral values or of a preprocessed version of the set of spectral values
2122
mapping values of a most-significant bitplane of the tuple of spectral values onto a group index and an element index, the element index designating an element within a group selected by the group index
2124
selecting a cumulative-frequencies-table out of a set of 32 cumulative-frequencies-tables in dependence on a state index describing a state of the arithmetic encoder
2126
arithmetically encoding the group index using a selected cumulative-frequencies-table, in order to obtain an arithmetically-encoded variable length-codeword

FIG 21

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/050954

A. CLASSIFICATION OF SUBJECT MATTER
INV. G10L19/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G10L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC, WPI Data, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MEINE NIKOLAUS ET AL: "IMPROVED QUANTIZATION AND LOSSLESS CODING FOR SUBBAND AUDIO CODING" PREPRINTS OF PAPERS PRESENTED AT THE AES CONVENTION, XX, XX, vol. 1-4, 31 May 2005 (2005-05-31), pages 1-9, XP008071322 * abstract page 1, left-hand column, paragraph 1 - right-hand column, paragraph 2 page 6, left-hand column, paragraph 3-4 page 6, right-hand column, paragraph 2-3 figures 5-7 ----- -/--	1-16

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 June 2010	06/07/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Greiser, Norbert

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/050954

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2007/066970 A1 (SAMSUNG ELECTRONICS CO LTD [KR]; MIAO LEI [KR]; OH EUN MI [KR]; KIM JU) 14 June 2007 (2007-06-14) * abstract page 1, paragraph [3] - page 3, paragraph [12] -----	1,13-16
A	"ISO/IEC 14496-3:2001(E) - Subpart 4: General Audio Coding (GA) . AAC, TwinVQ, BSAC" INTERNATIONAL STANDARD ISO/IEC, XX, XX, 1 January 2001 (2001-01-01), pages 1-313, XP007902531 page 3, paragraph [4.1 Scope] - paragraph [4.1.1.1 encoder and Decoder Block Diagrams] figures 4.1,4.2 -----	1,13-16

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2010/050954

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2007066970 A1	14-06-2007	NONE	